

AD-A135 899

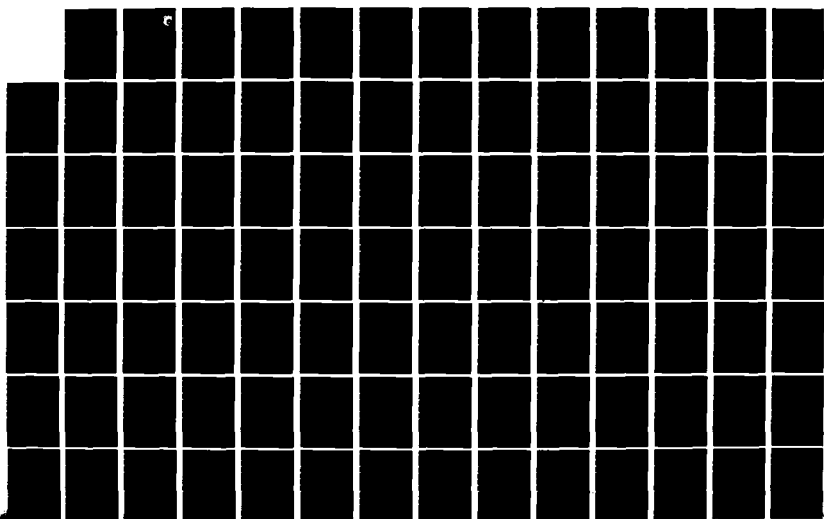
LSI (LARGE SCALE INTEGRATED) DESIGN FOR TESTABILITY
FINAL REPORT OF DESIG. (U) IBM FEDERAL SYSTEMS DIV
MANASSAS VA R D GROVES ET AL. NOV 83 AFMAL-TR-81-1068
F33615-79-C-1729

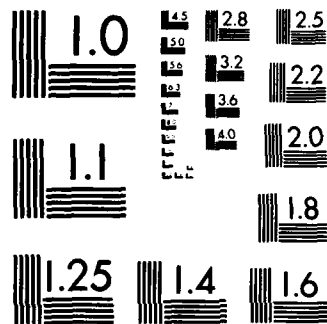
1/4

UNCLASSIFIED

F/G 5/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFWAL-TR-81-1068



LSI DESIGN FOR TESTABILITY

FINAL REPORT OF DESIGN, DEMONSTRATION, AND TESTABILITY ANALYSIS

IBM CORPORATION
FEDERAL SYSTEMS DIVISION
9500 GODWIN DRIVE
MANASSAS, VA 22110

NOVEMBER 1983

FINAL REPORT FOR PERIOD 5 JULY 1979 - 15 DECEMBER 1980

83 12 16 071
A

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DMC FILE COPY

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

83 12 16 071

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Thomas A. Herbert

THOMAS A. HERBERT, Acting Group Chief
Design & Fabrication Group
Microelectronics Branch

FOR THE COMMANDER

Stanley E. Wagner

STANLEY E. WAGNER, Branch Chief
Microelectronics Branch
Avionics Laboratory



"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MADE, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM								
1. REPORT NUMBER AFWAL-TR-81-1068	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER								
4. TITLE (and Subtitle) LSI DESIGN FOR TESTABILITY FINAL REPORT OF DESIGN, DEMONSTRATION, AND TESTABILITY ANALYSIS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL FINAL REPORT 05 JUL 79 - 15 DEC 80								
		6. PERFORMING ORG. REPORT NUMBER								
7. AUTHOR(s) Randall D. Groves Robert L. Schoenike		8. CONTRACT OR GRANT NUMBER(s) FY33615-79-C-1729								
9. PERFORMING ORGANIZATION NAME AND ADDRESS IBM Corporation, Federal Systems Division 9500 Godwin Drive Manassas, VA 22110		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Deliverable 0002-06 Final Report								
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Wright Avionics Lab AFWAL-AADE-3 Wright Patterson AFB, Ohio 45433		12. REPORT DATE November 1983								
		13. NUMBER OF PAGES 302								
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED								
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE								
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release, distribution unlimited.										
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)										
18. SUPPLEMENTARY NOTES										
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)										
<table border="0"> <tr> <td>Level Sensitive Scan Design</td> <td>Logic Testability</td> </tr> <tr> <td>LSSD</td> <td>Logic Design</td> </tr> <tr> <td>Design for Testability</td> <td>Test Design Constraints</td> </tr> <tr> <td>LSI Testability</td> <td>Test Generation</td> </tr> </table>			Level Sensitive Scan Design	Logic Testability	LSSD	Logic Design	Design for Testability	Test Design Constraints	LSI Testability	Test Generation
Level Sensitive Scan Design	Logic Testability									
LSSD	Logic Design									
Design for Testability	Test Design Constraints									
LSI Testability	Test Generation									
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)										
<p>The objective of this effort was to demonstrate IBM Level Sensitive Scan Design methodology as an approach for improving the testability of military LSI/VLSI circuits. LSSD was demonstrated in an LSI component AP101C test bed to be a viable and attractive design approach for military LSI/VLSI components.</p>										

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

		<u>Page</u>
	Executive Summary.....	xvi
1	<u>PROJECT OVERVIEW</u>	1
1.1	<u>INTRODUCTION</u>	1
1.2	<u>OBJECTIVE</u>	1
1.3	<u>SCOPE</u>	1
2	<u>RESULTS AND DATA OF DESIGN AND DEMONSTRATION</u>	3
2.1	<u>COMPARISON OF 74S481 AND SCS481</u>	4
2.1.1	74S481 BRIEF DESCRIPTION.....	4
2.1.2	DIFFERENCES BETWEEN 74S481 AND SCS481.....	4
2.1.2.1	<u>Slice Position Input</u>	6
2.1.2.2	<u>Pinout and Package</u>	7
2.1.2.3	<u>Power Supply Requirements</u>	7
2.1.2.4	<u>Electrical Requirements</u>	7
2.1.2.5	<u>Critical Path Performance</u>	8
2.1.2.6	<u>SCS481 Enhancements</u>	8
2.1.2.7	<u>SCS481 Test Circuits</u>	8
2.1.2.8	<u>SCS481 Design Errors and Corrections</u>	10
2.1.3	DESIGN DATA COMPARISON.....	10
2.2	<u>PRELIMINARY DEVICE EVALUATION</u>	12
2.2.1	MANUFACTURING TESTS (LSSD).....	12
2.2.2	MDS-800 EVALUATION.....	12
2.2.2.1	<u>MDS-800 Evaluation of Good SCS481's</u>	13
2.2.2.2	<u>MDS-800 Evaluation of Partially Good SCS481's</u>	13

TABLE OF CONTENTS (continued)

		<u>Page</u>
2.3	<u>TEST VEHICLE EVALUATION</u>	13
2.3.1	TEST VEHICLE EVALUATION OF GOOD SCS481's.....	13
2.3.2	TEST VEHICLE EVALUATION OF PARTIALLY GOOD SCS481's...	16
2.3.3	COMPARISON OF FTP AND MDS TESTS TO LSSD TEST.....	16
2.4	<u>COMPARISON OF LSSD AND NON-LSSD FAULT DETECTION</u>	20
2.4.1	LSSD TEST GENERATION RESULTS.....	20
2.4.2	NON-LSSD TEST GENERATION RESULTS.....	20
2.5	<u>SCS481 TESTABILITY</u>	23
2.5.1	ANALYSIS OF LATCH TESTABILITY.....	23
2.5.2	COMBINATIONAL LOGIC TESTABILITY.....	29
2.5.3	LSSD VS NON-LSSD TEST GENERATION COMPARISON.....	31
3	<u>DESIGN SUMMARY</u>	35
3.1	<u>LOGIC DESIGN</u>	37
3.1.1	MS399 TECHNOLOGY.....	37
3.1.2	LEVEL SENSITIVE SCAN DESIGN (LSSD) RULES.....	43
3.1.3	ALTERATIONS TO 74S481 FOR LSSD CONFORMANCE.....	50
3.1.3.1	<u>MS399 Shift Register Latches</u>	53
3.1.3.2	<u>On Chip Clock Generation</u>	53
3.1.3.3	<u>LSSD Rules Checking Program Violation Changes</u>	58
3.1.3.4	<u>Gated B-Clock Testability</u>	62
3.1.3.5	<u>LSSD Redesign Conclusions</u>	62
3.1.4	ENHANCEMENTS.....	64
3.1.4.1	<u>Bus From MC to A-Mux</u>	64
3.1.4.2	<u>Gate Clock to A, B Registers</u>	64
3.1.4.3	<u>4x4 Stack for PC</u>	66
3.1.4.4	<u>Breakpoint Register</u>	66
3.1.4.5	<u>Dual OP9 Outputs</u>	68
3.1.4.6	<u>B Bus Out Separated from Input</u>	68

TABLE OF CONTENTS (continued)

		<u>Page</u>
3.1.5	TEST CIRCUITS.....	68
3.1.5.1	<u>Ring Oscillator</u>	69
3.1.5.2	<u>DOP Three-state Control</u>	69
3.1.5.3	<u>CEE0 Test Circuit</u>	69
3.1.5.4	<u>CEC8/CEC0/CED8/CED7 Test Circuit</u>	69
3.1.6	LOGIC DESIGN ERRORS.....	69
3.1.7	DESIGN DATA COMPARISON.....	74
3.1.7.1	<u>Masterslice vs. Custom Implementation</u>	78
3.1.7.2	<u>LSSD Implementation Impacts</u>	78
3.2	<u>LOGIC ENTRY TO DATA BASE (LSSD)</u>	82
3.2.1	COLLECTION OF INDEXABLE DATA (CID).....	82
3.2.2	LOGIC BOOKS/MACROS.....	84
3.2.3	BASIC DESIGN LANGUAGE FOR STRUCTURE (BDL/S).....	84
3.2.4	INTERACTIVE SIMULATION AND ALD COMPOSING (ISAAC).....	89
3.2.5	AUTOMATED LOGIC DIAGRAM.....	90
3.2.6	MS399 STATISTICS PROGRAM.....	93
3.2.7	BDL/S SYNTAX CHECKER.....	95
3.2.8	SPECIAL LOGIC ENTRY REQUIREMENTS FOR I/O FLAGGING....	95
3.3	<u>LOGIC ENTRY TO DATA BASE (NON-LSSD)</u>	96
3.3.1	DFF BOOKS FOR NON-LSSD.....	97
3.3.2	NON-LSSD DESIGN WITH SETS AND RESETS.....	97
3.4	<u>CREATION AND ENTRY OF FUNCTIONAL PATTERNS</u>	100
3.4.1	BASIC DESIGN LANGUAGE FOR CONTROL.....	100
3.4.2	LOGIC PARTITION FOR SIMULATION.....	101
3.4.2.1	<u>Arithmetic Logic Operations</u>	101
3.4.2.2	<u>Control Logic</u>	102
3.4.2.3	<u>Address Registers</u>	102
3.4.2.4	<u>Data Register</u>	104
3.4.2.5	<u>Multiply, Divide, CRC</u>	104

TABLE OF CONTENTS (continued)

		<u>Page</u>
3.4.2.6	<u>Enhancements</u>	105
3.4.3	SUGGESTED IMPROVEMENTS TO DESIGN VERIFICATION.....	105
3.5	<u>DESIGN VERIFICATION (LSSD)</u>	107
3.5.1	EXPAND ACTION.....	107
3.5.2	SIMULATION.....	108
3.5.2.1	<u>Unit Delay Simulation</u>	110
3.5.2.2	<u>Nominal Delay Simulation</u>	110
3.5.2.3	<u>Calculated Delay Simulation</u>	111
3.5.3	ALL EVENTS TRACE (AET).....	111
3.5.4	TIMING ANALYSIS (TA).....	111
3.5.5	LSSD RULES CHECK.....	112
3.5.6	TESTABILITY ANALYSIS.....	112
3.5.7	LOGIC UPDATE.....	113
3.6	<u>DESIGN VERIFICATION (NON-LSSD)</u>	113
3.7	<u>PHYSICAL DESIGN</u>	115
3.7.1	PHYSICAL DESIGN TECHNOLOGY RULES.....	117
3.7.2	PHYSICAL DESIGN ACTIONS.....	117
3.7.3	MS399 PHYSICAL DESIGN.....	118
3.7.3.1	<u>CID Initialization</u>	120
3.7.3.2	<u>Physical Design Expansion</u>	120
3.7.3.3	<u>Placement</u>	121
3.7.3.4	<u>Wiring</u>	121
3.7.3.5	<u>Shapes Generation and Check</u>	121
3.8	<u>DELAY PATH VERIFICATION</u>	126
3.8.1	DELAY SIMULATION TIMING COMPARISON.....	126
3.8.2	PHYSICAL DESIGN CAPACITANCE CHECKS.....	130
3.8.3	AP-101C TIMING ANALYSIS.....	130
3.8.4	EFFECTS OF DESIGN ERROR CORRECTION CIRCUITS.....	144

TABLE OF CONTENTS (continued)

		<u>Page</u>
3.9	<u>TEST PATTERN GENERATION (LSSD)</u>	148
3.9.1	TEST GENERATION DEFINITIONS.....	148
3.9.2	EDS TEST GENERATION METHODOLOGY.....	151
3.9.3	TEST GENERATORS.....	154
3.9.3.1	<u>RANDGEN</u>	154
3.9.3.2	<u>FAULTGEN</u>	154
3.9.3.3	<u>PODEM</u>	155
3.9.3.4	<u>SOFTG</u>	155
3.9.4	LSSD TEST TECHNIQUES.....	155
3.9.5	LSSD TEST GENERATION PASS #1.....	156
3.9.6	LSSD TEST GENERATION PASS #2.....	159
3.9.7	LSSD TEST GENERATION COMPARISON.....	160
3.9.8	FAULT SIMULATION OF FUNCTIONAL PATTERNS.....	161
3.10	<u>TEST PATTERN GENERATION (NON-LSSD)</u>	167
3.10.1	SET/RESET NON-LSSD TEST GENERATION.....	167
3.10.2	STANDARD NON-LSSD TEST GENERATION.....	170
3.10.3	NON-LSSD TEST GENERATION COMPARISON.....	173
3.10.4	LSSD VS NON-LSSD UNTESTED FAULT COMPARISON.....	175
3.10.5	LSSD VS NON-LSSD TEST GENERATION CONCLUSIONS.....	181
3.11	<u>RELEASE DATA TO MANUFACTURING</u>	184
3.11.1	PRE-RELEASE CHECKING (AUDIT).....	184
3.11.2	INPUTS TO THE RELEASE PROCESS.....	185
3.11.3	PREPARING TEST DATA FOR RELEASE.....	186
3.11.4	PREPARING PHYSICAL DESIGN DATA FOR RELEASE.....	186
3.12	<u>FABRICATION AND TEST</u>	187
3.13	<u>TEST DIAGNOSTIC SELECTION</u>	188
3.13.1	FAILED DEVICE SELECTION.....	188
3.13.2	FAILED DEVICE ANALYSIS.....	188

TABLE OF CONTENTS (continued)

		<u>Page</u>
3.14	<u>TEST CIRCUIT BOARDS</u>	192
3.14.1	TEST CIRCUIT BOARD FOR 74S481.....	192
3.14.2	TEST CIRCUIT BOARD FOR SCS481.....	192
3.15	<u>MDS-800 TEST CONFIGURATION</u>	203
3.16	<u>DESCRIPTION OF TEST VEHICLE</u>	206
4	<u>RECOMMENDATIONS FOR LSSD DESIGN RULES IN A</u> <u>MILITARY ENVIRONMENT</u>	209
4.1	<u>LSSD RULE 1</u>	210
4.2	<u>LSSD RULE 2</u>	210
4.3	<u>LSSD RULE 3</u>	212
4.4	<u>LSSD RULE 4</u>	213
4.5	<u>LSSD RULE 5</u>	213
4.6	<u>LSSD RULE 6</u>	214
4.7	<u>LSSD RULES FOR TESTING INTERNAL ARRAYS</u>	215
4.8	<u>CONCLUSIONS</u>	219
5	<u>TESTER CHARACTERISTICS TO ACCOMMODATE LSSD</u>	221
5.1	<u>LSSD EFFECTS ON STANDARD TEST DATA VOLUME</u>	221
5.2	<u>TYPICAL LSSD TEST</u>	222

TABLE OF CONTENTS (continued)

		<u>Page</u>
5.3	<u>LSSD TESTER HARDWARE</u>	224
5.3.1	<u>RECURSIVE HARDWARE TO ACCOMMODATE LSSD</u>	224
5.3.1.1	<u>Recursive Controller</u>	226
5.3.1.2	<u>Recursive Memory</u>	229
5.3.1.3	<u>Programmable Timing of Recursive</u>	229
5.3.2	<u>SUMMARY</u>	232
6	<u>LSSD REQUIREMENTS FOR CAD SYSTEMS</u>	233
6.1	<u>SIMULATION</u>	233
6.1.1	<u>DESIGN RULES CHECKING</u>	233
6.1.2	<u>LSSD SRL MODEL</u>	233
6.2	<u>TEST PATTERN GENERATION</u>	234
6.2.1	<u>IDENTIFICATION OF CONTROLS</u>	234
6.2.2	<u>AUTO TEST PATTERN GENERATION</u>	234
6.2.3	<u>PATTERN SUBSUMPTION</u>	234
6.2.4	<u>FAULT SIMULATOR</u>	235
6.2.5	<u>DESIGN PARTITIONING</u>	235
6.3	<u>PHYSICAL DESIGN</u>	235
6.4	<u>DATA RELEASE</u>	235
7	<u>PROJECTION OF LSSD EFFECTS</u>	237
7.1	<u>CIRCUIT DENSITY</u>	237
7.1.1	<u>CIRCUIT DENSITY FOR SEMI-CUSTOM DESIGNS</u>	237
7.1.2	<u>CIRCUIT DENSITY FOR CUSTOM DESIGNS</u>	245
7.2	<u>PINOUT REQUIREMENTS</u>	245

TABLE OF CONTENTS (continued)

		<u>Page</u>
7.3	<u>LAYOUT COMPLEXITY</u>	247
7.4	<u>PERFORMANCE LIMITATIONS</u>	247
7.5	<u>POWER CONSUMPTION</u>	253
7.6	<u>LOGIC TESTER COMPLEXITY</u>	253
7.7	<u>TEST PATTERN GENERATION COSTS</u>	53
7.7.1	TEST PATTERN INTEGRITY.....	253
7.7.2	COMBINATIONAL LOGIC TESTING.....	254
7.7.3	AUTOMATIC PATTERN SUBSUMING.....	254
7.7.4	SIMPLIFIED PARTITIONING.....	255
7.7.5	SECOND LEVEL APPLICATIONS.....	256
7.7.6	CONCLUSIONS.....	256
7.8	<u>LIFE CYCLE COST</u>	257
7.9	<u>STANDARDIZATION</u>	257
7.10	<u>MAINTENANCE CONCEPTS</u>	258
7.11	<u>OPERATIONAL AVAILABILITY</u>	258
7.12	<u>RELIABILITY</u>	259
8	<u>IBM CONCLUSIONS AND RECOMMENDATIONS WITH RESPECT TO LSSD IN THE MIL ENVIRONMENT</u>	261
8.1	<u>CONCLUSIONS</u>	261
8.1.1	LSSD/NON-LSSD DESIGN.....	261
8.1.2	LSSD/NON-LSSD TEST CAPABILITY.....	261
8.2	<u>RECOMMENDATIONS</u>	262

LIST OF FIGURES

		<u>Page</u>
2.1.1-1	Functional Block Diagram for 'S481.....	5
2.1.2.6-1	SCS481 Functional Diagram.....	9
2.5-1	Polarity Hold (PH) SRL.....	24
2.5-2	Clocked Set/Reset (CSR) SRL.....	24
2.5-3	DFF #1 from Non-LSSD SCS481.....	25
2.5-4	DFF #2 from 74S481.....	25
2.5-7	Redundant Fault Example.....	30
3.0-1	EDS Chip Design Methodology.....	36
3.1-1	Physical Arrangement of MS399.....	39
3.1-2	Schematic of Schottky Current Switch.....	39
3.1-3	Schematic of In-phase Collector Dot.....	39
3.1-4	MS399 Shift Register Latch w/Clock Driver.....	41
3.1-5	Schematic of On-chip Receiver.....	39
3.1-6	Schematic of Off-chip Driver.....	39
3.1.2-1	Polarity Hold Latch.....	45
3.1.2-2	LSSD Logic Design.....	46
3.1.3-1	Standard 74S481 Design.....	51
3.1.3-2	LSSD 74S481 Design.....	51
3.1.3-3	Modified LSSD 74S481 Design.....	52
3.1.3-4	MS399 Dual Port SRL.....	54
3.1.3-5	On-chip Clock Generator.....	55
3.1.3-6	SCS481 Clock Generation Timing (Unit Delay).....	57
3.1.3-7	Actual SCS481 Timing.....	57
3.1.3-8	OP9 Feedback Problem.....	60
3.1.3-9	SRL Used to Break Feedback.....	60
3.1.3-10	Separate $\overline{\text{COUT}}$ for Divide.....	61
3.1.3-11	Separate OP9 into Two Pins.....	61
3.1.4-1	MC to A Mux Enhancement.....	65
3.1.4-2	4x4 Push/Pop Stack.....	65
3.1.4-3	Breakpoint Register.....	67
3.1.5-1	Ring Oscillator.....	70

LIST OF FIGURES (continued)

		Page
3.1.5-2	Three-state Control for DOP.....	71
3.1.5-3	CEEO Test Circuit.....	72
3.1.5-4	CEC8/CEC0/CED8/CED7 Test Circuit.....	73
3.2-1	Logic Entry/Simulation.....	83
3.2.2-1	Logical Representation.....	85
3.2-2	Automated Logic Diagram.....	91
3.2-3	MS399 Statistics Program Output.....	94
3 3-1	DFF to Replace CED6 SRL - D6SR.....	98
3.3-2	DFF to Replace CED5 SRL - D5SR.....	98
3.3-3	D6SR DFF with Set and Reset Lines.....	99
3.3-4	D5SR DFF with Set and Reset Lines.....	99
3.5-1	Unit Delay Simulation Output.....	109
3.6-1	Non-LSSD Unit Delay Simulation Output.....	114
3.7-1	Physical Design/Release.....	116
3.7-3	Placement Printout.....	122
3.7-4	Wiring Printout.....	123
3.8-4	Capacitance Delay for Net WR045CD60.....	133
3.8-5	Capacitance Delay for Net WR050CB60.....	134
3.8-6	Capacitance Delay for Nets WR225DH60, WR225CD60.....	135
3.8-7	CPU Execution Unit.....	136
3.8-8	AP-101C Clocking.....	138
3.8-9	AP-101C Multiply Timing (Worst Case).....	140
3.8-10	AP-101C Multiply Timing Actual Practice.....	142
3.8-11	AP-101C Multiply Timing Remaining Cycles.....	143
3.8-12	Initial Shift Bit Inversion Correction Circuitry.....	145
3.8-13	Circuits Required to Correct SCS481 Inversions.....	146
3.9.1-1	Stuck and Reduced Fault Examples.....	149
3.9.1-2	CEBO Test Models.....	149
3.9.2-1	Test Generation.....	152
3.9.2-2	Non-LSSD Auto Test Generator Approach.....	153
3.9.4-1	LSSD Auto Test Generator Approach.....	157
3.9.8-1	Manual Test Generation Methodology.....	162

LIST OF FIGURES (continued)

		<u>Page</u>
3.13-1	Failed Pattern Printout.....	189
3.13-2	Stuck Fault Failure Analysis Printout.....	191
3.14-1	74S481 Test Circuit Board Outline Drawing.....	193
3.14-2	74S481 Test Circuit Board Schematic.....	194
3.14-3	SCS481 Test Circuit Board Outline Drawing.....	197
3.14-4	SCS481 Test Circuit Board Schematic.....	198
4.7-1	General Structure of LSSD/Array Interface.....	216
5.2-1	Typical LSSD Product.....	223
5.3.1-1	LSSD Tester Hardware.....	225
5.3.1-2	Recursive Block Diagram.....	227
7.0-1	Edge-Triggered Design.....	238
7.0-2	Master-Slave Latch Design.....	239
7.0-3	Multi-Clock Design.....	240
7.1.1-1	LSSD Overhead in MS399.....	243
7.1.1-2	LSSD Overhead.....	244
7.4-1	Polarity Hold (PH) SRL.....	249
7.4-2	Clocked Set/Reset (CSR) SRL.....	249
7.4-3	DFF #1.....	250
7.4-4	Latch Timing.....	250

LIST OF TABLES

		<u>Page</u>
2.1.3-1	Design Data Comparison.....	11
2.2.2-1	MDS-800 Test of Partially Good SCS481's.....	14
2.3.2-1	Test Vehicle Evaluation of Partially Good SCS481's...	18
2.4-1	Test Generation Results.....	21
2.5-5	Latch Testability Analysis.....	26
2.5-6	Untested Faults Left in Latches.....	27
2.5.3-1	Fault Coverage by Location.....	32
3.1.2-3	Summary of LSSD Design Rules.....	48
3.1.7-1	TI 74S481 to IBM SCS481 Comparison.....	76
3.1.7-2	Base Function-Technology Efficiency Comparison.....	77
3.1.7.2-1	Design Data Comparison.....	81
3.4-1	BDL/C Dataset Description for Simulation.....	103
3.7-2	MS399 Physical Design Steps.....	119
3.7-5	MS399 Design Statistics.....	125
3.8-1	54S/SCS481 Critical Path Performance Comparison.....	127
3.8-2	Clock Driver Net Capacitances.....	131
3.8-3	Capacitance Delay Impact.....	132
3.9.5-1	LSSD Auto Test Generation Comparison.....	158
3.9.8-2	BDL/C Data Sets for Manual Test Generation.....	163
3.9.8-3	Fault Simulation.....	165
3.10-1	Set/Reset Non-LSSD Test Generation Steps.....	168
3.10-2	Standard Non-LSSD Test Generation Steps.....	171
3.10.4-1	Fault Comparison.....	176
3.10.4-2	Net Fault Breakdown.....	178
3.10.4-3	Net Undetected Fault Analysis.....	179
3.10.4-4	Fault Coverage by Location.....	180
3.10.4-5	Combinational Test % to Redundancy Comparison.....	182
3.15-1	MDS-800 Test Description.....	205
7.4-5	SRL vs DFF Performance Comparison.....	251

APPENDIX

		<u>Page</u>
A	<u>SCS481/74S481 DIFFERENCES</u>	A-1
A.1	<u>SLICE POSITION INPUT</u>	A-1
A.2	<u>SCS481/74S481 PINOUT COMPARISON</u>	A-2
A.3	<u>SCS481/74S481 PACKAGE COMPARISON</u>	A-6
A.4	<u>POWER SUPPLY REQUIREMENTS</u>	A-7
A.5	<u>SCS481/74S481 ELECTRICAL REQUIREMENTS</u>	A-8
A.6	<u>CRITICAL PATH PERFORMANCE COMPARISON</u>	A-9
A.7	<u>SCS481 ENHANCEMENTS</u>	A-10
A.8	<u>SCS481 TEST CIRCUITS</u>	A-15
A.9	<u>SCS481 LOGIC DESIGN ERRORS AND CORRECTIONS</u>	A-18

EXECUTIVE SUMMARY

OBJECTIVES

In the spring of 1978 Air Force Wright Aeronautical Laboratories (AFWAL) Avionics Laboratory wanted to determine if there was a uniform design methodology capable of being generally applied to LSI designs (1000 gates) that would assure highly testable product in the first design pass. A highly testable product was deemed to be a definite requirement as LSI components have increasingly populated military electronics systems modules and have brought attendant increases in spares requirements and life cycle costs. After a review of approaches to testable design in use at that time AFWAL selected IBM's Level Sensitive Scan Design (LSSD) approach for a trial redesign of a well-defined component in a standard contractual data processing product. The objective was to compare that LSSD design of this component to the non-LSSD design and demonstrate the interchangeability of this level sensitive device in an edge triggered environment after actual manufacture of the device. An over-all evaluation of LSSD as it applies to the military environment was also included in the effort.

APPROACH

The approach to achieving these objectives was to redesign the Texas Instruments 74S481 four-bit slice microprocessor to conform with the LSSD design ground rules. The implementation technology was IBM's 1500 gate Schottky current switch masterslice MS399. The redesigned part was designated the SCS481 and was then substituted for the least significant slice 74S481 in the B-52 G/H AP-101C Offensive Avionic System computer to demonstrate functionality.

BACKGROUND

In the LSSD technique, the only type of storage element (other than arrays) permitted in a logic design is called a shift register latch (SRL). The SRLs, built with level sensitive polarity hold latches, are connected

together into a long shift register for testing purposes. This enables the integrated circuit test system to load data into and retrieve data from any storage element on the chip by mean of a simple shift technique. Only a few additional structured design rules must be followed to assure a highly testable design.

SPECIFICS

The SCS481 was successfully designed and implemented in a single pass using 1242 gates. The design was accomplished using IBM's Engineering Design System (EDS) CAD Software. Test patterns were generated automatically by the EDS LSSD test generation system. In order to provide a comparison, a non-LSSD version of the SCS481 was also created using DFF's in place of the LSSD SRL's. Test patterns were generated for the non-LSSD SCS481 using the EDS non-LSSD test generator.

The LSSD design achieved better test coverage, especially in the sequentially complex stack, while requiring significantly less CPU time and manual effort. This better coverage was achieved at an overall cost of 3.5% additional gates and 6.5% additional pins. This percentage overhead required for LSSD was considered to be typical for mapping LSI edge triggered designs into LSSD.

In general LSSD was found to have no significant performance impact when compared to edge-triggered designs. In fact, the LSSD SRL is capable of avoiding layout sensitive "fast path" problems associated with DFF's.

LSSD SRL's were found to be more testable than their DFF counterparts. They greatly enhance testability since they are essentially additional primary inputs and primary outputs for testing purposes (that is, data can be shifted into or read out of the latches by the test system).

The LSSD SCS481 successfully functioned in the edge-triggered environment of the AP-101C computer by passing the AP-101C Factory Test Program. Additionally, twenty-eight partially good SCS481's were also tested in the

AP-101C. Seven of those twenty-eight passed all AP-101C tests, however, only one device had a fault located within the 74S481 functions used by the AP-101C. This fault demonstrated characteristics associated with a physical short between the two metal levels of the SCS481.

CONCLUSIONS

LSSD appears to be a very viable approach for improving LSI/VLSI testability in the military environment. This increased testability can be achieved with relatively minor additional circuits and I/O pins. Further investigation of applying LSSD-like guidelines to second-level designs using vendor SSI/MSI TTL should be included in follow on work associated with this contract.

SECTION 1 PROJECT OVERVIEW

1.1 INTRODUCTION

In supporting its new-generation data processing products with Large Scale Integrated (LSI) Technology IBM has developed special procedures and techniques which will provide enhanced equipment reliability and serviceability. These techniques are transparent to equipment function, serve to "structure" the logic design approach, and enhance defect screening before product assembly and test.

These techniques are collectively known as the Level Sensitive Scan Design (LSSD) methodology and are employed in the "Purdue" logic technology (700 Cells, 3ns, 4.6 mm chip) first announced as the LSI technology for System 38, October 24, 1978.

1.2 OBJECTIVE

The objective of this contract effort was to demonstrate and evaluate IBM's LSSD methodology as an approach for significantly improving the testability of LSI and Very Large Scale Integrated (VLSI) circuits in military applications. The LSSD concept involves use of special latches which perform as shift register stages in addition to the normal storage latch function in LSI circuits. These Shift Register Latches (SRLs) give the capability for chaining all the latches on the chip which, when combined with a structured design philosophy, permits ~ 100% fault coverage of the logic design with automated test pattern generation.

1.3 SCOPE

Eight work scope elements were accomplished under this contract;

- (a) Design and simulation of component employing LSSD concept
- (b) Component layout and test pattern generation

- (c) Wafer processing, test and evaluation
- (d) Component packaging and testing
- (e) Preparation of demonstration vehicle functionality exercise
- (f) Benefits analysis of LSSD
- (g) Preparation and submission of documentation
- (h) Delivery of sample hardware to AFWAL

These elements were accomplished, ahead of schedule and under budget, by implementing a LSSD version of the TI 74S481 bit slice microprocessor using IBM's master slice technology MS399. Since the logic circuits in MS399 are implemented with Schottky Current Switches, the resulting design is called the SCS481. The SCS481 contains 1242 logic gates, 86 I/O's and performs flawlessly within the 74S481 least significant slice position in the B52 AP101C demonstration computer.

The SCS481 has become an LSI design case model for IBM Federal System Division and is being considered for use in succeeding IBM FSD equipment designs.

Further details of design, test evaluation and functional verification are contained in the balance of this report.

Since the base contract was accomplished in "one-pass" (vs. a "two-pass" plan), and the results of the base contract are very encouraging, IBM and AFWAL are discussing an extension of the base Statement of Work to include applications of LSSD to second level packages having mixed mode (LSSD with non-LSSD) logic and memory components.

SECTION 2

RESULTS AND DATA OF DESIGN AND DEMONSTRATION

This section contains a brief description of the purpose of this contract, a comparison of the 74S481 and the SCS481, and the results of good and bad device operation in the test vehicle as well as the results of the LSSD and non-LSSD testability study. Detailed analysis of the results including a discussion of design methodology are contained in Section 3.

The objective of this effort was to investigate and demonstrate IBM's Level Sensitive Scan Design (LSSD) methodology as an approach for significantly improving the testability of LSI and Very Large Scale Integrated (VLSI) circuits.

The LSSD concept involves use of special level sensitive latches which perform as shift register stages during testing in addition to the normal storage latch function in LSI circuits. These shift register stages give the capability for chaining all the latches on the chip which, when combined with a structured design philosophy, permits approximately 100% fault coverage of the logic design with automated test pattern generation. The approach used in completing the objective was to design and build in IBM technology MS399, the logical equivalent of a TI 74S481 but with LSSD test capabilities in place. This device is known as the Schottky Current Switch 481 (SCS481). The device was built and proven operational in a single pass through a normal chip development cycle. A test vehicle, IBM's AP-101C computer, that uses the 74S481 was selected to demonstrate the operation of the SCS481.

The yield from building integrated circuits is never 100%. Some of the partially good devices were selected, packaged, and tested to determine where the errors existed. These devices were then demonstrated in the test vehicle with the intent of proving the superiority of LSSD test methods.

2.1 COMPARISON OF 74S481 AND SCS481

2.1.1 74S481 BRIEF DESCRIPTION

The TI 74S481 is a four bit slice, bipolar, microprogrammable micro-processor. The TI 74S481 offers the logic designer great flexibility in achieving cost effective hardware designs. It has the following specific features:

- (a) Clock cycle time of 90 nS
- (b) Microprogrammable, bit-slice design expandable in 4-bit multiples
- (c) Full parallel dual input/output ports for use in advanced memory-to-memory architecture
- (d) Full-function ALU with carry look-ahead, magnitude, and overflow decision capabilities
- (e) Double-length accumulator with full shifting capability and sign-bit handling
- (f) Dual memory address generators on-chip
- (g) Simultaneous one-clock compound operations, with status.
- (h) Pre-programmed CRC and double-precision multiply/divide algorithms.
- (i) Double length accumulator with full bidirectional single/ double precision arithmetic/logical/circulate shift capabilities include sign protection.
- (j) 1.9 Watts power dissipation

The functional diagram for the 74S481 is shown in Figure 2.1.1-1.

2.1.2 DIFFERENCES BETWEEN 74S481 AND SCS481

The SCS481 design was implemented in IBM's MS399 technology. MS399 is a 1496 logic gate LSI bipolar masterslice with 94 TTL compatible I/O signal lines. The internal gates consist of three input OR/NOR gates which operate on an externally supplied 1.7 volts with a nominal delay of 1.5ns. High power versions of the gates are available with a nominal delay of 0.8ns.

SN74S481

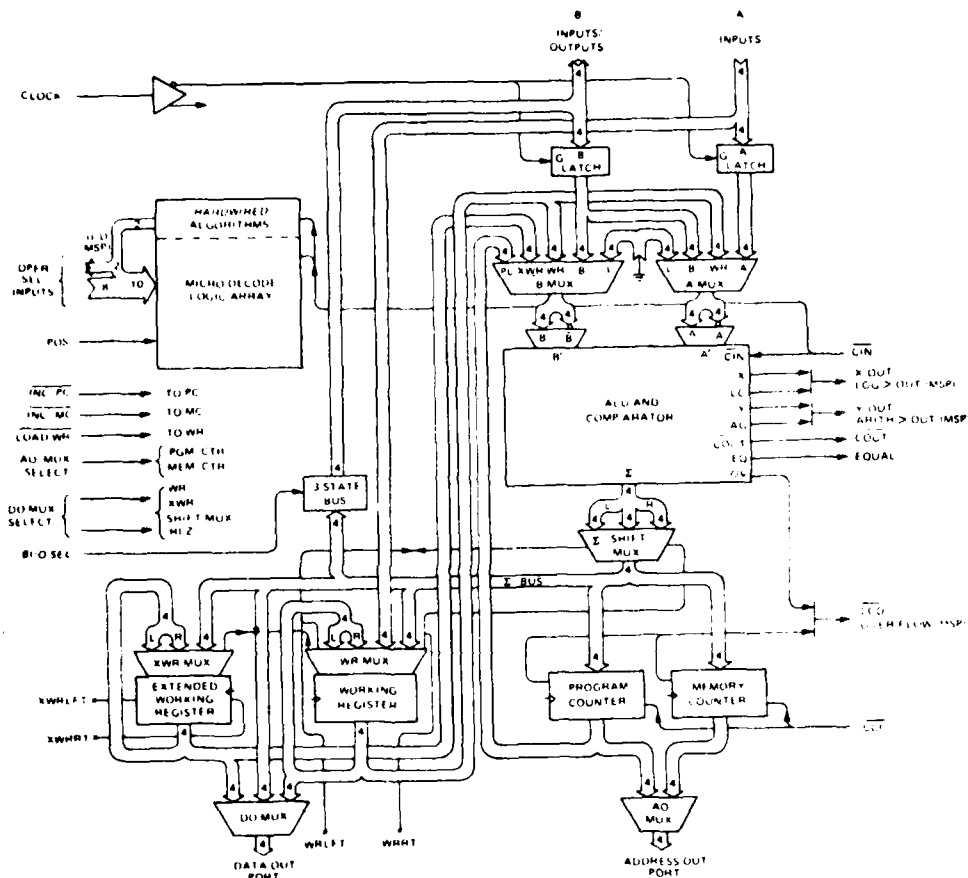


Figure 2.1.1-1 FUNCTIONAL BLOCK DIAGRAM FOR 'S481

From TI Catalog

Either gate output can be collector-dotted to form an AND function or to construct polarity hold latches. The MS399 chip is packaged in a 28mm ceramic carrier with an 11x11 array of pins. The chips are manufactured in IBM's East Fishkill facility in Hopewell Junction, NY. More details are provided in Section 3.1.1.

Although the SCS481 performs the functions of the 74S481, minor differences exist between the two parts since different technologies, packaging and designs were used for their implementation. These differences exist in the following areas:

- (a) Slice Position Input
- (b) Pinout
- (c) Package
- (d) Power Supply Requirements
- (e) Electrical Requirements
- (f) Critical Path Performance
- (g) SCS481 Enhancements
- (h) SCS481 Test Circuits
- (i) SCS481 Logic Design Errors and Corrections

2.1.2.1 Slice Position Input

Operation of the 74S/SCS481 is different depending on the location of the device in the hierarchy of the system. This location, least significant position (LSP), intermediate position (IP), or most significant position (MSP), is determined by the analog level on a single pin in the 74S481. The position input on the SCS481 is described digitally on two pins because the necessary analog comparator circuits were not available in MS399.

74S481

- (a) PIN 19 AT 0V -- IP SLICE
- (b) PIN 19 AT 2.4V -- LSP SLICE
- (c) PIN 19 AT 5V -- MSP SLICE

SCS481

- (a) PINS D2, L9 = 00, MSP SLICE
- (b) PINS D2, L9 = 11, LSP SLICE
- (c) ELSE = , IP SLICE

2.1.2.2 Pinout and Package

Pinout differences exist in that the 74S481 is packaged in a Quad-In-Line 48 pin package where the SCS481 is packaged in a carrier that has an array of 11 x 11 pins. The 5 pins in the center are not used resulting in 116 available pins. The pinout and package differences are detailed in Appendix A. Please note that all three-state outputs in the 74S481 are implemented as open collector outputs in the SCS481 except the DOP and B Bus out which are implemented as active drivers.

2.1.2.3 Power Supply Requirements

The SCS481 dissipates 1.4 watts of power versus 1.9 watts for the 74S481. The following line items describe the power supply requirements:

74S481

- (a) + 5V \pm 5% AT 380 mA TYP
- (b) 1.9 WATTS TYP

SCS481

- (a) + 5V \pm 10% AT 53 mA TYP
- (b) + 1.7V \pm 5% AT 670 mA TYP
- (c) 1.4 WATTS TYP.

2.1.2.4 Electrical Requirements

Due to the unique attributes of the two technologies, differences exist in input and output characteristics for the 74S481 and the SCS481. However, all SCS481 I/O characteristics are TTL compatible. A complete description of the electrical requirements for the SCS481 are included in Appendix A.

2.1.2.5 Critical Path Performance

The critical path performance comparison is discussed in Section 3.8.1 and Appendix A. The delays for the 74S481 were obtained from the TI catalog. The delays for the SCS481 were obtained from a delay simulation program. No SCS481s were physically measured for propagation delay. However, all calculated delays for the SCS481 met or exceeded the TI specifications.

2.1.2.6 SCS481 Enhancements

Unused cells and logic I/O in the MS399 gate array after the initial SCS481 design allowed six enhancements to be added to the 74S481 functions. A functional block diagram of the enhanced SCS481 is shown in Figure 2.1.2.6-1. These enhancements are documented in Section 3.1.4:

- (a) Bus from MC to A-Mux
- (b) Gate the clock to the A,B Registers
- (c) 4 x 4 Stack for the PC
- (d) Breakpoint Register
- (e) Dual OP9 outputs
- (f) B-Bus Out Separated from Input

2.1.2.7 SCS481 Test Circuits

Since extra logic and extra I/O pins existed on the SCS481 after the logic design of the 74S481 functions, four basic test circuits were added. These four circuits are described in Section 3.1.5:

- (a) Ring Oscillator
- (b) DOP Three-state Control
- (c) CEE0 Test Circuit
- (d) CEC8/CEC0/CED8/CED7 Test Circuit.

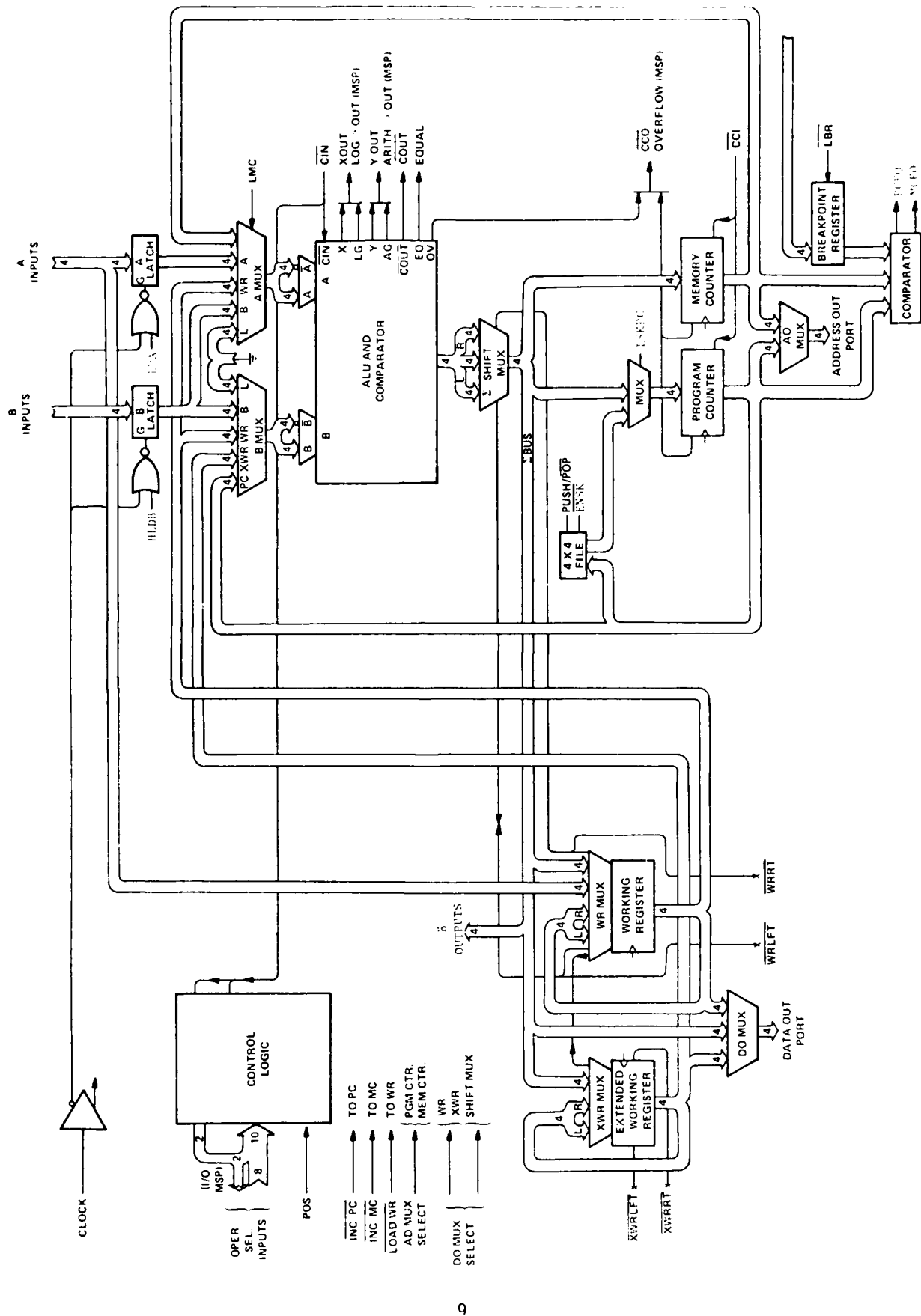


Figure 4.1.2.6-1 SCS481 Functional Diagram

2.1.2.8 SCS481 Logic Design Errors And Corrections

Logic design errors resulted in five I/O lines being logically inverted from their function in a 74S481. Those lines are XWRRRT, WRRRT, XWRLFT, WRLFT, and CCI (inverted for LSP). The circuits required to correct these errors are discussed in Section 3.8.4 and detailed in Appendix A. These circuits were mounted on the SCS481 test circuit board and used in demonstrating its capability.

2.1.3 DESIGN DATA COMPARISON

A summary of the gate and pin count comparisons between the SCS481 and the 74S481 is included in table 2.1.3-1. A description of how these numbers were arrived at is found in Section 3.1.7. Comparing the 74S481 (1) to the non-LSSD SCS481 (2) indicates the approximate cost associated with implementing this design in a masterslice instead of a custom approach. These 88 additional gates amount to a 12.2% increase in gate count, most of which occurred in the map over of the microcode decode PLA to the discrete NOR logic.

A comparison of the non-LSSD SCS481 (2) to the LSSD SCS481 (3) shows the minimum cost incurred in MS399 for implementing the 74S481 design in LSSD. The LSSD gate count amounted to 28 additional gates or 3.5%. Three pins or 6.5% were required to implement LSSD.

Row four of Table 2.1.3-1 shows the gate and pin count for the 74S481 function as actual implemented in the SCS481. The additional pins required are for implementing the position input digitally (Section 2.1.2.1), B-Bus output separation, and other unneeded LSSD pins (Section 3.1.3.5). The reasons for the additional gates are explained in Section 3.1.7.

Rows five and six of table 2.1.3-1 provide a comparison of the minimum implementation of a non-LSSD and LSSD SCS481 with enhancements. The LSSD cost incurred in this case is 0.5% for gates and 3.9% for pins. The decrease is primarily due to the fact that the LSSD overhead is now spread over a larger number of gates and pins.

TABLE 2.1.3-1 Design Data Comparison

	Gates	Signal Pins
1. 74S481	723(est.)	46
2. Non-LSSD SCS481 (base function)	811	46 @
3. LSSD SCS481 (base function)	839*	49 *@
4. LSSD SCS481 (as implemented in target technology)	1120	59
5. Non-LSSD SCS481 + Enhancements	1063	77
6. LSSD SCS481 + Enhancements	1068*	80 *
7. Complete SCS481 (as implemented in target technology)	1307	86
Masterslice (1) vs Custom Design (2) Cost	12.2%	0%
LSSD vs. non-LSSD Design Cost		
* (2vs3) '481 function minimum implementation in MS399	3.5%	6.5%
* (5vs6) SCS481 function minimum implementation in MS399	0.5%	3.9%

*These gate and pin counts assume the minimum LSSD requirements including off chip clock generation

@These pin counts do not include the additional pin required for the digital position input

The final gate and pin count for the SCS481 denotes the number of pins and gates used in the actual implementation of the SCS481. The differences between row six and seven are explained in detail in Section 3.1.7.

2.2 PRELIMINARY DEVICE EVALUATION

The test and evaluation of IBM's SCS481 consisted of device gate level testing in IBM's manufacturing facility in East Fishkill, NY using LSSD test patterns generated by IBM in Manassas, VA. A gross functional test was performed on an MDS-800 test station in Manassas before final evaluation in the AP-101C test vehicle in Owego, NY.

2.2.1 MANUFACTURING TESTS (LSSD)

All SCS481 chips which were processed through the manufacturing line at IBM's East Fishkill location were subject to LSSD test patterns to determine whether they had been manufactured properly. These patterns were generated at IBM Manassas using the EDS test generation methodology described in Section 3.9.

As a result of these tests, fourteen sample modules which passed all tests were sent for evaluation. An additional 100 good modules were later ordered for delivery to the customer and for additional evaluation. By examining the data from these tests, twenty-eight modules which failed at least one LSSD test were selected for comparison with the good modules (Section 3.13).

2.2.2 MDS-800 EVALUATION

When the SCS481's were first received from manufacturing, they were initially tested at IBM Manassas using an MDS-800. A description of the test set up and test circuit boards is provided in Section 3.14 and 3.15. The results of the MDS-800 evaluation on both the good and partially good SCS481's are detailed below.

2.2.2.1 MDS-800 Evaluation of Good SCS481s

The good SCS481s and their test circuit board were connected to the MDS-800 and exercised. After some minor wiring and power supply problems were corrected, the SCS481's successfully passed the MDS-800 tests.

2.2.2.2 MDS-800 Evaluation of Partially Good SCS481s

Twenty-eight partially good SCS481s were tested on the MDS-800 and data tabulated on them. These twenty-eight had previously been LSSD tested in East Fishkill and the probable function on the chip containing the error noted. Of the twenty-eight devices with known errors, seven passed the MDS-800 tests completely. Six devices passed one of the two basic MDS-800 tests. Fifteen devices failed both MDS-800 tests. The results of this testing along with the probable area of the fault within the device (determined from LSSD tests in East Fishkill) is shown in Table 2.2.2-1.

2.3 TEST VEHICLE EVALUATION

The test vehicle (AP-101C) and its factor test program (FTP) are described in detail in section 3.16. The results from evaluating good and partially good SCS481's in this test vehicle are detailed below.

2.3.1 TEST VEHICLE EVALUATION OF GOOD SCS481s

A good SCS481 and its test circuit board were connected to the test vehicle and the FTP was run. Additional ground connections were required between the SCS481 test circuit board and the AP-101C logic page for proper function. These ground wires were added and FTP was successfully executed except for the multiply test which requires high speed operation of the shift line, XWRRT. The inverter circuits originally developed to correct the shift lines were discovered to be inadequate in speed for the multiply operation. To prove this, the AP-101C master oscillator (20MHz) was disconnected and a lab oscillator (variable frequency) connected in its place.

Table 2.2.2-1 MDS-800 Test of Partially Good SC5431's

DEVICE	FAILED LSSD TEST PATTERN #	MDS TESTS PASSED	# FAILURES IN MDS TESTS		PROBABLE NET CAUSING FAILURE
			1	2	
1	667	0	27	19	WR MUX Bit 3
2	67	1	0	8	XWR Bit 3 s-a-0, pattern sensitive (Chip Join Error)
3	608	0	22	13	A Bus Bit 3 In
4	1175(1518)	2	0	0	OP Form VII (Compare), Equal (Chip Join Error)
5	42	1	1	0	COUT s-a-0 for MSP generate
6	1325	2	0	0	XWR, Single Precision Arithmetic Shift in MSP
7	135	0	28	1	Test Circuits and Stack
8	74	0	25	27	
9	41	0	7	17	
10	139	0	2	29	Logical Operation in ALU
11	44	0	18	5	
12	186(688)	2	0	0	Sum Bus Right Shift Arithmetic MSP (Chip Join Error)
13	667	0	31	7	WR MUX Bit 2
14	676	2	0	0	OP Form XI/XIV Hold XWR (Signed Div/Mult)
15	8	0	19	4	
16	794	2	0	0	Right Double Precision Shift with Sum Bus in MSP
17	669	0	24	21	XWR MUX Bit 1
18	684	1	0	2	Mult/Div Decode, OP 9 Out, XWRLFT s-a-0

Table 2.2.2-1 MDS-800 Test of Partially Good SCS481's (Continued)

DEVICE	FAILED LSSD TEST PATTERN #	MDS TESTS PASSED	# FAILURES IN MDS TESTS		
			1	2	
19	171	1	0	5	WRLEFT s-a-1, Sum Bus Bit 3 s-a-0
20	55	0	28	23	
21	667	1	27	0	WR MUX SEL 1 OP Form I
22	57	0	23	18	
23	6	0	6	3	
24	127	2	0	0	Test Circuits or Stack
25	279	2	0	0	OP Form II Sum Bus Shift
26	72	1	0	5	
27	674	0	27	23	WR MUX Controls s-a-1
28	57	0	3	11	

With the slow inverter circuits, the entire FTP was executed successfully with the lab oscillator set at 18.3 MHz. The inverter circuit described in section 3.8.4 was then designed and added to the SCS481 test circuit board. With the new active signal inverters, the FTP was successfully executed with the lab oscillator set at 20.5 MHz. The AP-101C 20MHz master oscillator was connected back into the system and the FTP successfully executed on 13 good SCS481's. This demonstrated the successful operation of a level sensitive device (SCS481) in an edge triggered environment (AP-101C).

2.3.2 TEST VEHICLE EVALUATION OF PARTIALLY GOOD SCS481's

Twenty-eight partially good SCS481s (whose faults were described in Table 2.2.2-1) were tested on the test vehicle (AP-101C). Of the seven parts that passed all MDS tests, five passed all AP-101C tests. The remaining two passed all but one AP-101C test. Of the six parts that passed one MDS test, four failed all AP-101C tests but two passed all AP-101C tests. Of the fifteen parts that failed both MDS tests, all fifteen failed the AP-101C tests. The detailed results of the testing is shown in Table 2.3.2-1.

2.3.3 COMPARISON OF FTP AND MDS TESTS TO LSSD TEST

The fifteen partially good SCS481's which failed both functional tests require no explanation; however, the seven which passed the MDS tests and the seven which passed the AP-101C FTP tests need further examination. First, looking at the five chips which passed both the MDS and FTP tests shows that their faults are in non-functional areas of the chips. Chip #4 has faults for the EQ signal in OP Form VII. Neither the MDS nor the FTP use this OP Form. Chips #6, #12, and #16 all have errors detectable only in the MSP shifts. Since the 74S481 in use is configured as an LSP, these are non-functional faults. Chip #24 has fault(s) in the test circuits and/or stack which do not even exist in the 74S481.

The two chips which passed on the MDS but not the FTP are chips #14 and #25. Both these chips have faults in OP Forms not executed by the MDS but which are used by the AP-101C, namely OP Forms XI and II respectively.

These OP Forms are used infrequently on the AP-101C explaining why these two parts were able to pass all the FTP tests but one.

The two chips that passed the FTP but failed on the MDS are more interesting. Chip #5 has an error in the COUT circuit which cause COUT to be low even when the two most significant bits of the operands are both zero. Since a carry look ahead generator ('182) is used in the AP-101C instead of a ripple carry, the COUT line in the LSP is non-functional.

The fault specified by the analysis program in East Fishkill for Chip #2 was the XWR most significant bit s-a-0 (stuck at zero). However, this fault does not exhibit standard stuck fault characteristics. Several MDS and FTP tests demonstrate that a one can be successfully loaded and read out of XWR bit 3. Both the MDS and LSSD tests show that certain patterns cause this one to "magically" disappear tending to indicate that some sort of intermetal level short exists. By chance, the AP-101C FTP does not ever detect this situation.

Even though the auto test generators were not attempting to test the fault which exists in Chip #2 because it can not be modeled as a stuck fault, the argument can be made that the greater than 97% test coverage achieved increased the chances that LSSD test patterns would detect it. In fact this fault was detected during the Shift Register test. The FTP tests, with their less comprehensive test coverage, were less fortunate than the LSSD tests in detecting this fault.

For this small sample of parts, LSSD test patterns have no clear advantage over the manually generated patterns in those areas of the chip they both exercise. LSSD was able to detect a fault which the manual patterns did not; but, since this fault is not a classic stuck fault, no firm conclusion about LSSD's effectiveness can be made. More than likely, the stuck fault coverage of the LSSD test is superior to the manual pattern coverage, but no convenient comparison exists. Most importantly, the LSSD tests took less than an hour of CPU time to generate and debug; whereas, the MDS and FTP tests took many times that number of man-hours to reach the same condition with probably inferior test coverage.

Table 2.3.2-1 Test Vehicle Evaluation of Partially Good SCS411's

SCS411	SCS TESTS PASSED	AP-101C						TEST OPERATION FAILED
		22EE BASIC	259E BRANCH	2F68 ARITH	354C LOGICAL	3BA6 SHIFT	41EC BYTE	
1	0	F						Won't Run; Fails compare test
2	1	P	P	P	P	P	P	
3	0	F						Won't load PC; loads 8, not 0
4	2	P	P	P	P	P	P	
5	1	P	P	P	P	P	P	
6	2	P	P	P	P	P	P	
7	0	F						Won't load H/W
8	0	F						Modifies instruction
9	0	F						Fails store instruction
10	0	F						Fails compare test
11	0	F						Won't reset
12	2	P	P	P	P	P	P	
13	0	F						Won't load H/W
14	2	P	P	F	P	P	P	CPUART 12 fails; mult H/W
15	0	F						Reg (4000 x 4000)
16	2	P	P	P	P	P	P	Fails load H/W
17	0	F						Fails load PC
18	1	F						Fails shift left logical
19	1	F						Fails load MC
20	0	F						Won't reset
21	1	F						PC dropping bit

Table 2.3.2-1 Test Vehicle Evaluation of Partially Good SCS481's (Continued)

SCS481	MDS TESTS PASSED	API01C FTP						TEST OPERATION FAILED
		22EE BASIC	259E BRANCH	2F08 ARITH	354C LOGICAL	3BA6 SHIFT	41EC BYTE	49A6 SPEC
22	0	F						PC load problem
23	0	F						PC load problem
24	2	P	P	P	P	P	P	CPUART 15 fails; Improper Multiply correction
25	2	P	P	F	P	P	P	Fails Shift left logical Won't reset
26	1	F						Fails Shift left logical
27	0	F						Won't reset
28	0	F						Fails Shift left logical

H/W - Halfword

2.4 COMPARISON OF LSSD AND NON-LSSD FAULT DETECTION

In order to provide a comparison of the testability of LSSD and non-LSSD design, two non-LSSD versions of the SCS481 were entered into IBM's Engineering Design System (EDS) and test generation was performed. Test generation methodology for LSSD and non-LSSD designs in EDS differs significantly. These differences are described in section 3.9. The results of test generation on both the LSSD and non-LSSD SCS481's are detailed in the following sections.

2.4.1 LSSD TEST GENERATION RESULTS

Two separate test generation runs were made on the LSSD design. The first attempt was as a testability analysis as described in Section 3.5.7. At this point the decision was made to insert a ring oscillator and several test circuits into the logic. The design, including test circuits, then proceeded through physical design and another test generation run. The results from these two test generation runs are shown in Table 2.4-1.

The two attempts yielded fairly similar results with both above 99% tested; however, the computer time for the second run is nearly twice that of the first run. This additional CPU time can be attributed to the serendipity involved in using random patterns. Note that neither job required any substantial manual effort other than job submission and retrieval.

2.4.2 NON-LSSD TEST GENERATION RESULTS

Table 2.4-1 also shows the results from two non-LSSD test generation runs. In both cases, the LSSD logic was altered by deleting the scan path and replacing the SRL's with edge-triggered DFF's. In one case, set and reset lines were added to the DFF and all were tied together to their respective set and reset primary input pin. In order to normalize CPU time a maximum time limit of 56 CPU minutes was set.

Table 2.4-1 Test Generation Results

	LSSD#1	LSSD#2	Non-LSSD	Non-LSSD w/sets & resets
Test Percentage	98%	97.4%	95.9%	89.8%
Net Test Percentage	99.9%	99.1%	96.8%	88.25%
Patterns Generated	1761	2275	5365	2794
Saved CPU Minutes	31:49	56:03	56:18	56.19
Total CPU Minutes Used	31:49	56:03	92:10	66:19
Labor-Man Hours	.2	.2	16	10

Both non-LSSD test generation runs achieved good results. Even though the standard non-LSSD version achieved better final results than the set/reset version, additional manual intervention and wasted CPU time was required. The standard non-LSSD design also required nearly twice as many patterns. Obviously the set/reset SCS481 was found to be easier to test by the EDS test generators.

Direct extrapolation of this test coverage to that which would be achieved by the original TI design should be made cautiously. As explained in Sections 3.1.3 and 3.3, several changes to the design were made which might have improved the testability. These changes involve the use of a more testable DFF model, the use of the \bar{Q} output of that model, and the break apart and other changes to the B-Bus, OP8, and OP9 circuitry.

2.5 SCS481 TESTABILITY

Given that this particular LSSD design is more testable than the non-LSSD versions, an understanding of where and why these test coverage differences exist is important in understanding what circuit characteristics caused the non-LSSD test generator to have difficulty. The best place to begin this analysis is to study in detail the testability of the individual latch designs.

2.5.1 ANALYSIS OF LATCH TESTABILITY

The elementary gate representations of the two most popular versions of the basic shift register latch (SRL), the polarity hold (PH) SRL, which is the version used in the SCS481, and the clocked set/reset (CSR) SRL, are shown in Figures 2.5-1 and 2.5-2 respectively. A logical representation of two edge-triggered delay flip-flops (DFF) are shown in Figure 2.5-3 and 2.5.4. DFF version one was used in the non-LSSD design. DFF version two was actually used in the 74S481 design. Adding set and reset lines to either DFF does not affect the theoretical testability of the latch; so, for clarity, those designs will not be discussed here.

The testability of the DC stuck faults found in the four latch types is discussed in Table 2.5-5. The PH SRL is completely testable in all configurations which conform to LSSD Rules. This is substantiated by the fact that all latch faults were completely tested by the LSSD test patterns. The CSR SRL has four stuck faults which are untestable if the L1 and $\overline{L1}$ outputs are not used. The equivalent faults for the scan input and the L1 to L2 connection are testable during the scan path flush test in which both the A and B clocks are turned on simultaneously. These four faults are collapsed or reduced to two faults for test generation. The use of both the L1 and $\overline{L1}$ outputs in conformance with LSSD rules usually results in 100% testability.

DFF #1 has one fault which is always untestable, input 3C s-a-0. Table 2.5-6 shows that this fault was untested for all 46 latches in both non-LSSD designs. If Q does not feed through exclusively combinational logic to a

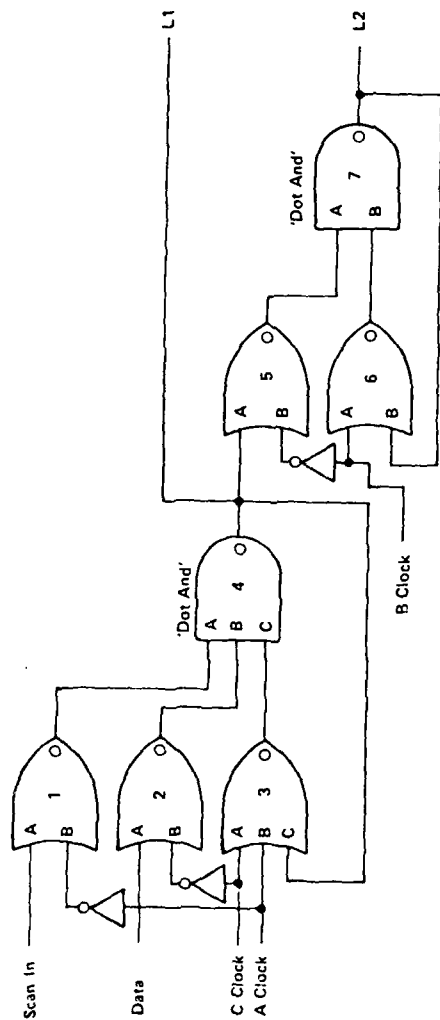


Figure 2.5-1 Polarity Hold (PH) SRL

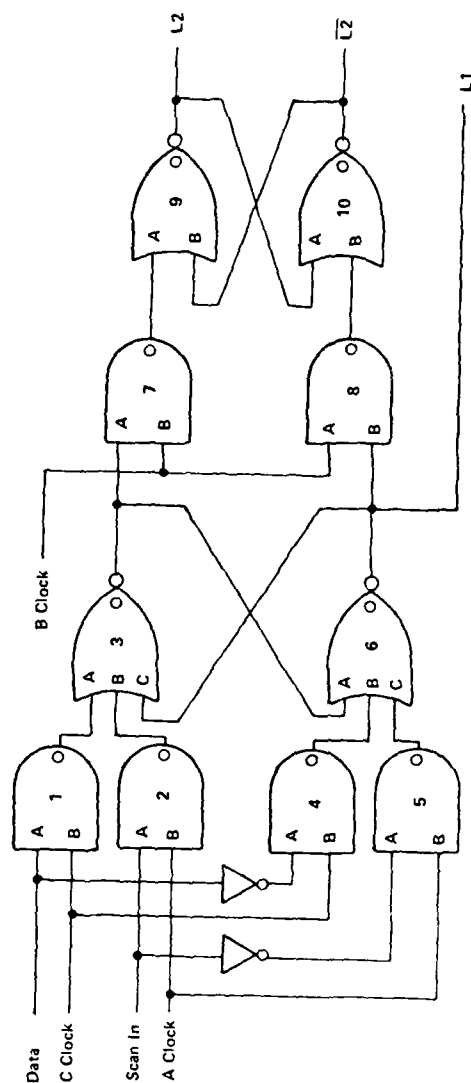


Figure 2.5-2 Clocked Set/Reset (CSR) SRL

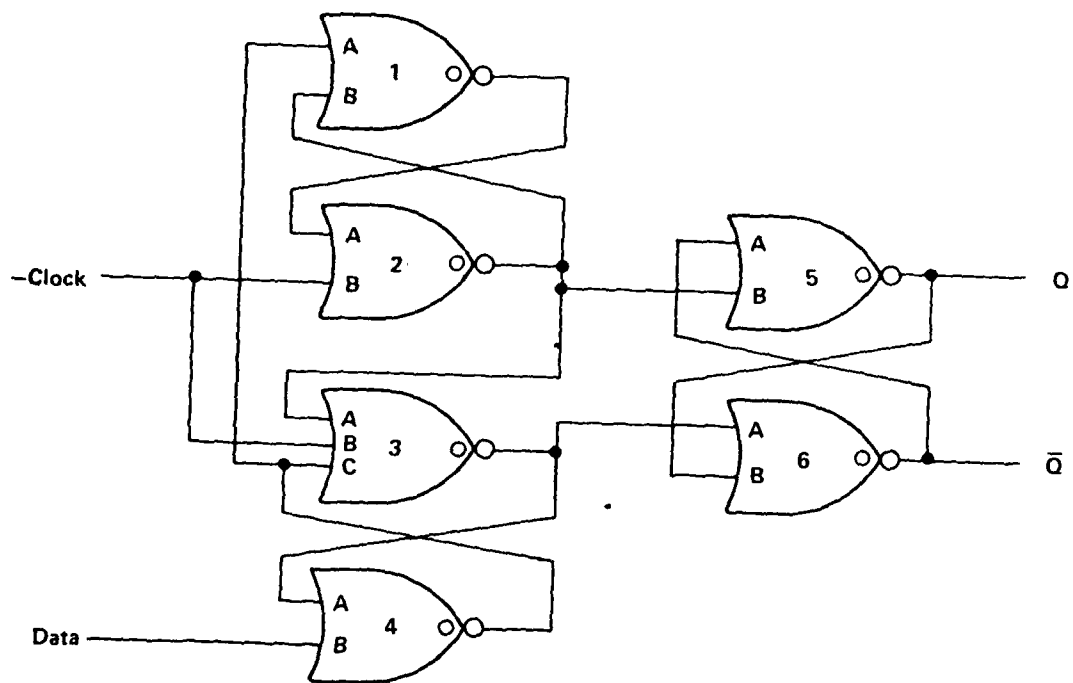


Figure 2.5-3 DFF#1 from non-LSSD 50S481

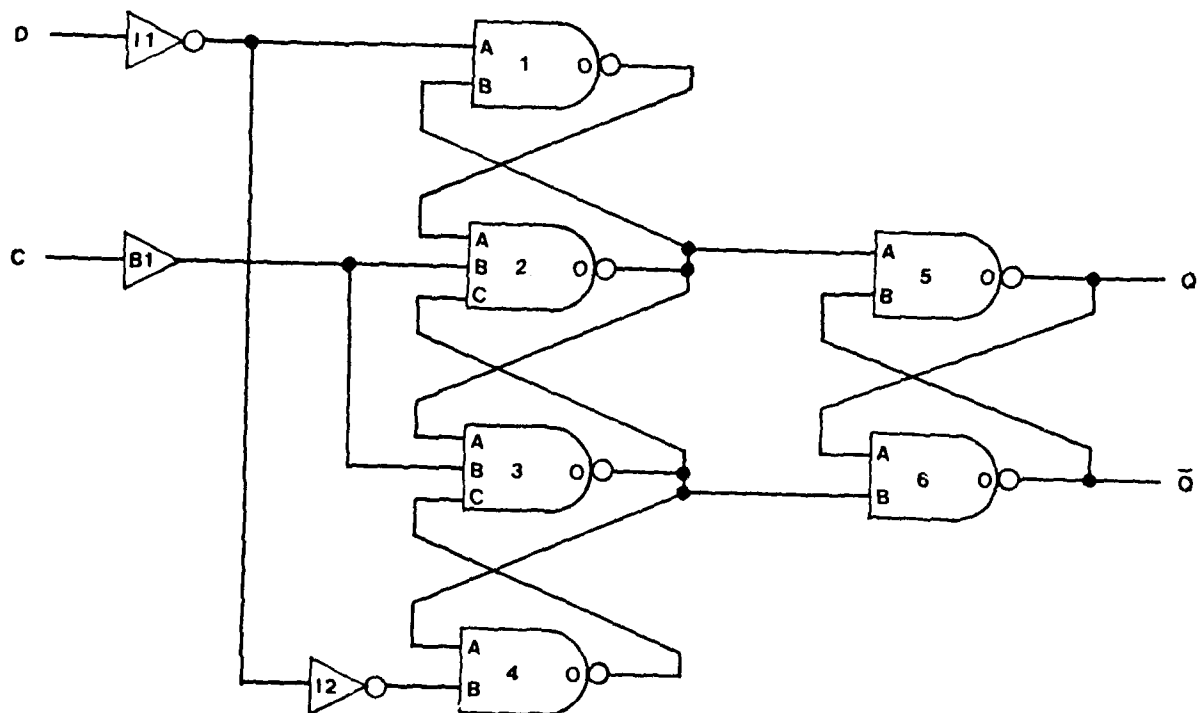


Figure 2.5-4 DFF#2 from 74S481

TABLE 2.5-5
LATCH TESTABILITY ANALYSIS

1. Polarity Hold SRL - 100% Testable
2. Clocked Set/Reset SRL-4 DC Stuck Faults Not Testable
 - Input 1A S-a-1
 - Input 4A S-a-1
 - Data Inverter Input S-a-0 and Output S-a-1

The equivalent faults in the scan path and L2 are tested during the flush test of the scan string. These 4 faults become testable if L1 output is used according to LSSD rules.

3. Edge Triggered DFF #1-2 DC Struck Faults Not Testable
 - Input 3A S-a-0
 - Input 3C S-a-0

Input 3A S-a-0 can be detected if \bar{Q} goes to a primary output through combinational logic only or goes to another DFF which is clocked differently. Input 4A S-a-0 causes automatic test generator problems when the DFF is imbedded in sequential logic.

4. Edge Triggered DFF #2-6 DC Stuck Faults Not Testable
 - Input 2A S-a-1 and Output 10 s-a-1
 - Input 3C S-a-1 and Output 40 s-a-1
 - Data Inverter 2, Input S-a-1 and Output S-a-0

Inputs 1B, 2C, 3A, and 4A s-a-1 cause automatic test generator problems when imbedded in sequential logic.

Table 2.5-6. Untested faults left in the latches

LATCH * FAULT	NON-LSSD w/S/R Untested Faults						NON-LSSD w/o S/R Untested Faults					
	Non Stack	Stack Ø	Stack 1	Stack 2	Stack 3	TOTAL	Non Stack	Stack Ø	Stack 1	Stack 2	Stack 3	TOTAL
1A			1	2	4	7						-
B			1	2	4	7	1					1
C			4	4	-	8						-
2A						-						-
B		1	1	4	4	10						-
C		3	3	3	-	9						-
3A		4	4	4	4	16	1	4	4	4	4	17
B						-						-
C	22	4	4	4	4	38	22	4	4	4	4	38
+ 4A		2	2	4	4	12	2				1	3
B						-						-
C			4	4	4	12					1	1
D	6	4	4	4	4	22						-
5A						-						-
B	3		2	4	4	13						-
C	1	4	4	4	-	13						-
6A		1	1	4	4	10						-
B						-						-
C					1	1						-
+ 7A		1	1	4	4	10			2	2	4	8
B					1	1						-
C		1	1	4	4	10						-
D		4	4	4	4	16						-
8A						-						-
B						-						-
30s-a-Ø		1	1	4	4	10						-
TOTAL	32	30	42	63	58	225	26	8	10	10	14	68

* See Figures 2.5-3 and 3.3-2

+ Input 7A s-a-Ø in Figure 3.3-2 is analagous to input 4A s-a-Ø

primary output or feed another latch using a different clock controllable from a primary input, 3A s-a-0 becomes untestable. The latch output used in the non-LSSD designs is essentially an inverted \bar{Q} (see Figure 3.3-1) and all non-stack latches go exclusively through combinational logic to a primary output. This means 3A s-a-0 is theoretically testable for all non-stack latches. Excepting one latch, the only 3A s-a-0 faults left untested in both non-LSSD designs were in the stack (see Table 2.5-6).

DFF #2 has six faults which are always untestable. Due to the nature of DFF's, no potential circuit utilizations will make these faults testable. These six faults are reduced to three for test generation.

DFF #1 has one fault which is theoretically testable, but causes automatic test generators some difficulty. Input 4A s-a-0 creates this problem because its detection requires the data input to change from a 1 to a 0 with the clock still active. DFF #2 has similar problems with faults 1B, 2C, 3A, and 4A s-a-1 which require the data input to change both directions with an active clock. If the DFF is fed from other sequential logic, this means the preceding latches must be set up to change the data input on the rising edge of the clock. Therefore, the test generator must be concerned about setting up the data input of more than one latch. If the DFF is buried in several layers of sequential logic, the decision tables for the test generation algorithm will probably overflow in attempting to test these faults. Table 2.5-6 shows that twenty-two of these faults remained untested in the set/reset non-LSSD version and eleven faults were untested in the standard non-LSSD version. Since DFF #2 has four faults of this type, one can safely assume that if it had been used instead of DFF #1, the test percentages would have been even lower.

The PH SRL is obviously superior from a testability standpoint, however, the latch does tend to glitch on the rising edge of the clock. Glitchless versions of the PH SRL are possible, but they introduce an untestable fault which can only be detected as a glitch. Glitches are seldom a concern in LSSD designs anyway, unless that design feeds asynchronous non-LSSD logic. The CSR SRL has a greater number of untestable faults, but in many applications they can become completely testable. DFF #1 has fewer potential

untestables than the CSR SRL, but input 3C s-a-0 is always untestable and the glitches caused by it could create problems in an edge-triggered design. DFF #2 has the most untestables; and they never become testable under any circumstances. Additionally 4A s-a-0 for DFF #1 and 1B, 2C, 3A, and 4A s-a-1 for DFF #2 will always give test generators difficulty.

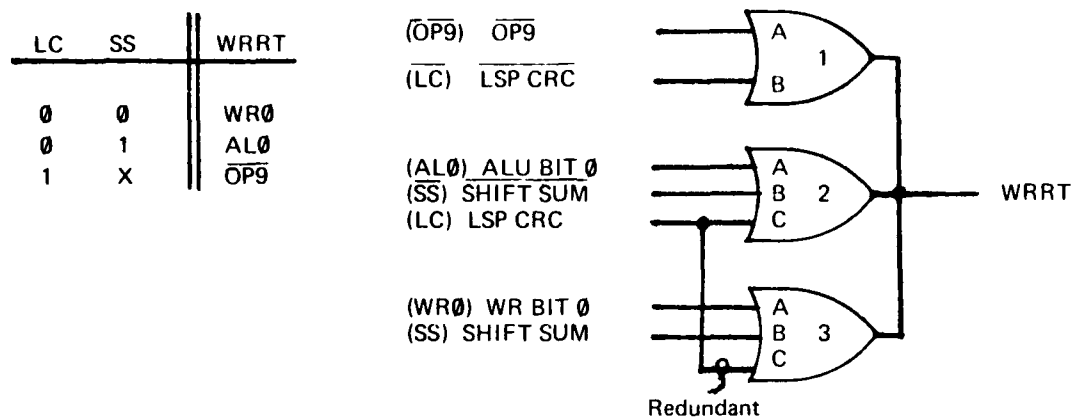
2.5.2 COMBINATIONAL LOGIC TESTABILITY

The combinational logic section of the SCS481 design is common between all three versions of the design. In converting TI's custom NAND logic to IBM masterslice NOR logic, 126 redundant faults (17 reduced) were introduced into the combinational logic. By definition, redundant logic is unnecessary for the proper DC function of the device. Examination of the redundant logic resulted in the conclusion that all of the faults could easily be removed with six percent circuit savings and improved testability. For the purposes of net test percentage calculations, the redundant faults have been removed from both the total faults and any untested faults in the design.

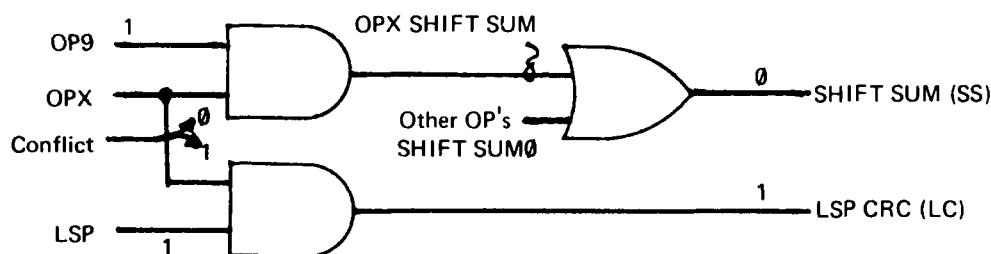
An example of a redundant fault is shown in Figure 2.5-7(a). The WRRT multiplexor shown in the figure is the implementation of the truth table on the left. The Boolean equation for the mux (Figure 3.5-7(c) equation 1) using LC and SS is logically non-redundant; however, a redundant fault exists. In order to discover why, the combinational logic which generates LC and SS must be examined.

Figure 2.5-7(b) represents a simplified version of this combinational logic. The necessary conditions to test fault 3C s-a-0 are shown. Propagating LC and SS back through the combinational logic creates a conflict since the net OPX must be 1 and 0 simultaneously.

Writing a non-redundant Boolean equation for the WRRT mux using $\overline{OP9}$, OPX and LSP results in equation (2) in Figure 3.5-7(c). Substitution of equations (3) and (4) into equation (1) results in equation (5). A comparison of equation (2) and (5) reveals the redundant term which is equivalent to LC. Therefore, deleting LC from 3C would not change the DC result at WRRT.



(a) Redundant fault in MUX



Testing 3C s-a-0 requires: $SS=0, WR0=0, LC=1, \overline{OP9}=1$

(b) Combinational LOGIC driving MUX

$$(1) \quad WRRT = (\overline{LC} + \overline{OP9})(\overline{SS} + LC + AL)(SS + LC + WR)$$

$$(2) \quad WRRT = (\overline{OPX} + \overline{LSP} + \overline{OP9})(OP9 + \overline{OPX} + LSP + AL)((\overline{OP9} \cdot \overline{OPX}) + WR)$$

$$(3) \quad LC = OPX \cdot LSP$$

$$(4) \quad SS = OPX \cdot \overline{OP9}$$

$$(5) \quad WRRT = (\overline{OPX} + \overline{LSP} + \overline{OP9})(OP9 + \overline{OPX} + LSP + AL)((\overline{OP9} \cdot \overline{OPX}) + (OPX \cdot LSP) + WR)$$

Redundant[†] Term = LC

(c) Boolean Equations

Figure 2.5-7 Redundant Fault Example

Redundant logic faults are untestable using a single stuck fault model since detection of a redundant fault would require more than one fault to be present. The untestable latch faults just discussed are untestable, but they are not redundant. They are necessary for the proper operation of the latches; therefore, they have not been removed for net test percentage calculations.

2.5.3 LSSD VS NON-LSSD TEST GENERATION COMPARISON

Table 2.5.3-1 shows a detailed breakdown of test coverage for the LSSD and non-LSSD test generation efforts. The lack of sequential depth in the standard 74S481 design simplifies automatic test generation since over 70% of the faults are combinational in nature. This fact is corroborated by the similar test coverage percentages obtained in both the LSSD and non-LSSD combinational faults. However, as pointed out in Section 3.3, direct comparison of the testability of the SCS481 non-LSSD designs to the 74S481 non-LSSD design is not completely applicable since potential enhancements to the testability of the SCS481 versions were included.

Even though the non-LSSD test coverage figures compare favorably with the LSSD coverages, the non-LSSD attempts are prone to chance and involve more learning intervention, and computer time. This additional effort differentiates LSSD and non-LSSD test generation. This difference tends to increase exponentially with increasing sequential complexity. Comparing the non-LSSD efforts with LSSD pass #1 provides a sharp contrast between the two approaches. In less than 32 minutes, basically 100% of the faults were detected. By linear extrapolation, both non-LSSD test generation attempts tested less than 65% of the logic after 32 minutes.

LSSD's advantages become obvious in the latch fault coverage, especially in the stack logic. These advantages accrue from both the greater testability of the level sensitive PH SRL and the scanability of these latches. Coupled with the benefits of total automatic test pattern generation and subsumption, the fact that an in depth design testability analysis is unnecessary - provided the design rules are followed - and that the

TABLE 2.5.3-1 FAULT COVERAGE BY LOCATION

LOCATION	ALL FAULTS				COMBINATIONAL FAULTS				CLOCK/LATCH FAULTS			
	NON		S/R NON		NON		S/R NON		NON		S/R NON	
	LSSD 1	LSSD 2	LSSD	LSSD	LSSD 1	LSSD2	LSSD	LSSD	LSSD 1	LSSD 2	LSSD	LSSD
ALU	99.8	99.5	99.1	95%	100	99.4	99.4	95.3	99%	100	94.1	92.3
ADDRESS REGISTERS	100	98.3	97.7	84.4	100	97.5	99.5	77.9	100	100	93.7	96.3
DATA REGISTERS	100	99%	95.5	93.9	100	98.7	96.5	94.2	100	100	93.1	93.2
MACRO LOGIC	100	96.6	97.9	81%	100	95.9	98.5	78.6	100	100	95.2	90.4
INSTRUCTION DECODE	100	100	98.9	93%	100	100	98.9	93%	-	-	-	-
NON-STACK												
ENHANCEMENT	99.8	99.3	95.3	95%	100	99.4	98.9	93.9	99%	99%	88.2	96.7
STACK	99.3	99.3	88.9	66.3	100	100	66.7	55.6	99.3	99.3	89.3	66.4
TOTAL	99.9	99.2	96.8	88.3	100%	99%	98.7	91.1	99.5	99.6	91.2	81.6

technique is universally applicable to all designs from the simplest to the most complex, LSSD provides significant advantages for design testability, especially as sequential complexity and gate to pin ratios increase.

SECTION 3

DESIGN SUMMARY

The following is a description of the experiences encountered during the design and fabrication of the SCS481. Included are the details of logic design and entry as well as the particulars on the use of the Engineering Design System (EDS) for design verification, test generation, physical design, and release to manufacturing. A description of the test circuit boards and the test setups for the MDS-800 and test vehicle, the AP-101C, are also contained in the concluding sections. The differences between the LSSD and non-LSSD designs during logic entry, design verification, and especially test generation will be discussed in detail.

EDS Chip Design Methodology Overview - The general EDS chip design methodology flow is shown in Figure 3.0-1. The entire process begins with the design of the logic to be placed on the chip. This design is then entered into the data base and preliminary technology checking is performed. The design is then simulated to verify function and any logic corrections are entered into the data base. This process of simulation and logic update is continued until the designer is satisfied with the design and simulation results.

Physical design consists of placement and wiring of the components used in the design as well as technology checking and generation of the information used to make masks. In parallel, the logic designer uses information from physical design to verify that the circuit delays determined by "actual" physical design information are appropriate for the design. Any potential problems can then be corrected in physical design.

Upon conclusion of physical design, the data base passes on to test generation. There, patterns are generated to test the design. Complete diagnostic information is created. This information is combined with the physical design data and a final audit check is made of the entire process. A release tape is then generated and sent to the manufacturing facility.

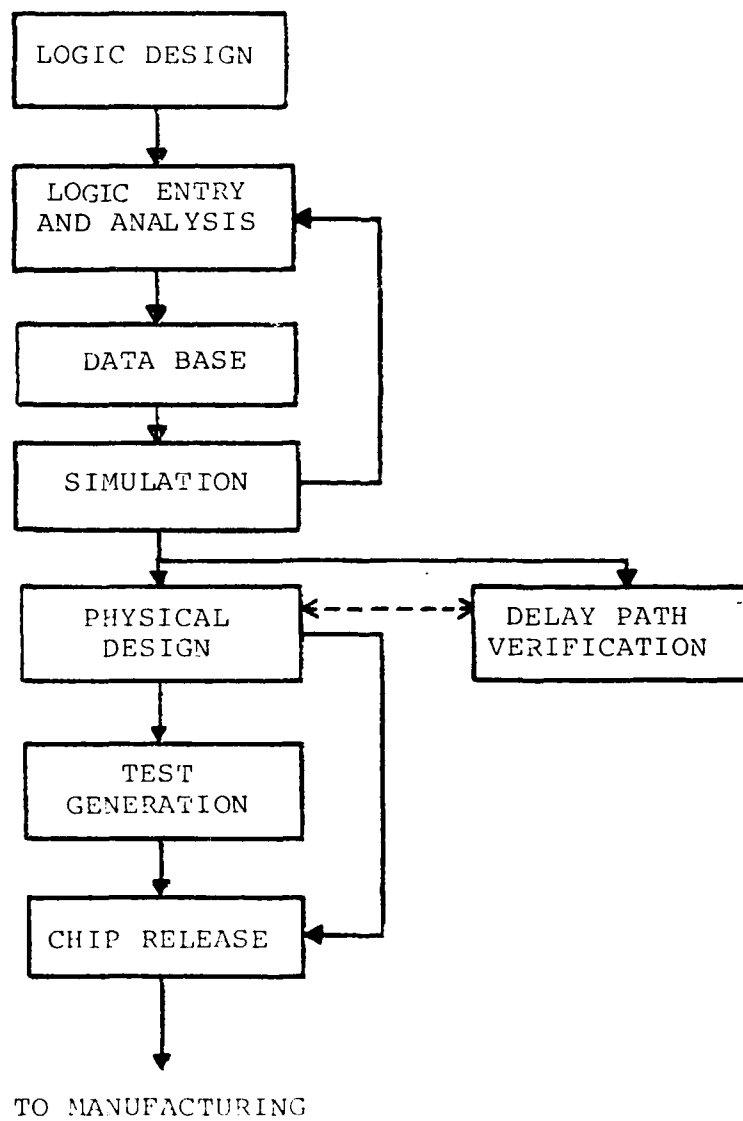


Figure 3.0-1 EDS Chip Design Methodology

3.1 LOGIC DESIGN

IBM Manassas IRAD task 9M66 selected as the LSSD machine vehicle for this contract the Texas Instruments 74S481 four bit bit-slice ALU to be implemented in IBM technology MS399 using LSSD design rules. The 74S481 is described in Section 2.1.1. Being a well defined function with logic diagrams available to IBM from TI, the logic design effort primarily involved converting those custom TTL NAND gate diagrams to a master slice NOR gate implementation with an eye on assuring conformance to LSSD rules. The resulting differences between the designs are documented in Appendix A. The reasons for these differences are detailed in the following subsections.

3.1.1 MS399 TECHNOLOGY

The IBM MS399 technology has been described in recent literature*. Some excerpts from that article are included here. MS399 is a bipolar Schottky-clamped emitter coupled logic technology arranged in a master slice or gate array configuration. The standard logic gates are connected together in a custom manner using the various metal levels on the chip. The chip itself is 5.66 mm square, with an active area of 5.43 mm² and a 0.23 mm² kerf. Contained within this active area are 1496 logic gates, 88 receiver circuits, 60 high-power driver circuits, and six reference generators. Up to 94 I/O signal lines are available, with T²L compatible voltage levels. The chip requires ground and two power supply voltages: +5.0V \pm 10.0 percent (V_{CC}), and 1.70V \pm 5.0 percent (V_C). Overall power dissipation on the chip is minimized by using the lower voltage to power the internal logic gates and on-chip receivers, while the higher voltage is used to power the off-chip driver and reference generator circuits. The exact power dissipation will vary with the number of circuits wired, but will nominally run 1.8W. The chip will operate over a range of junction temperatures from -55°C to 125°C.

*Al500 Gate, Random Logic, Large-Scale Integrated (LSI) Masterslice, R.J. Blumberg & S. Brenner, IEEE Journal of Solid State Circuits, Vol. SC-14, No. 5, Oct. 1979, pp. 818-822.

provided the temperature cycling between the extremes is kept to a minimum (a new military package must be developed before this technology can be used in a military environment).

The chip is designed around an array of internal logic cells which are arranged in 34 rows and 22 columns, for a total of 748 cells. Up to two circuits per cell can be wired, hence the total 1496 internal logic gates. The level-shifting receiver and driver circuits, as well as the reference generating circuits, are located along the periphery of the chip (see Figure 3.1-1). Each cell is provided with an optimum amount of space for power and signal lines to enter and exit. The first two levels of metal and the interconnecting or via level are used to custom wire or personalize the masterslice. Circuit utilization can be fairly high over the entire range of part numbers and can run in excess of 90 percent.

Figure 3.1-2 is the schematic diagram of the internal logic gate. It is a standard emitter-coupled logic (ECL) or current switch circuit, with no emitter followers. The circuit performs the OR/NOR logic function, with both phases available. The availability of both phases can result in a significant savings, over a single-phase technology, in the number of circuits needed to implement a specific logic function. An up logic level is equal to $+V_c$ volts or 1.70V. A down level is equal to $(V_c - V_{fSBD})$ volts, which results in a 600 mV signal swing on-chip. V_r , the reference potential which is generated on chip, is located midway between the up and down levels. The V_c supply voltage was chosen to be as low as possible (in order to minimize chip power dissipation), but high enough to keep all junctions on and conducting under worst case conditions.

The internal circuit comes in two versions. The first operates at an average current of 0.5 mA, and is the predominant circuit used on any part number. The second version utilizes the center taps in the resistors and operates at an average current of 1.0 mA. This high power circuit is used where performance is critical or when a high fan-out capability is required. For the low-power circuit, the fan-out is limited to five low-power or two high-power circuits. This restriction prevents the up level from being

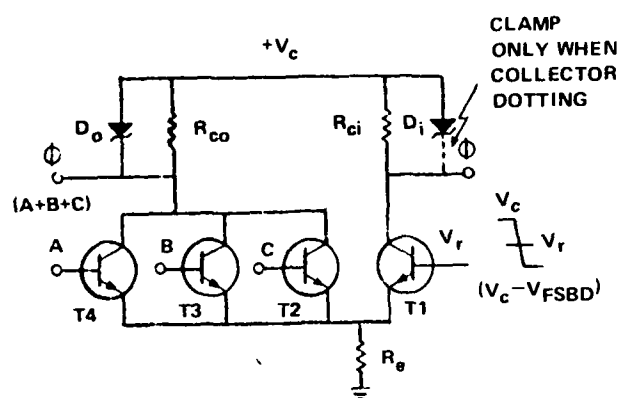


FIGURE 3.1-2 Schematic diagram of Schottky diode clamped current switch logic circuit.

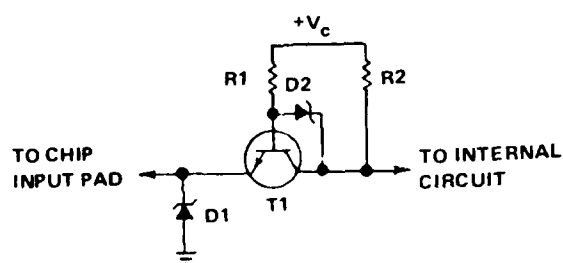


FIGURE 3.1-5 Schematic diagram of on-chip receiver level shifting circuit.

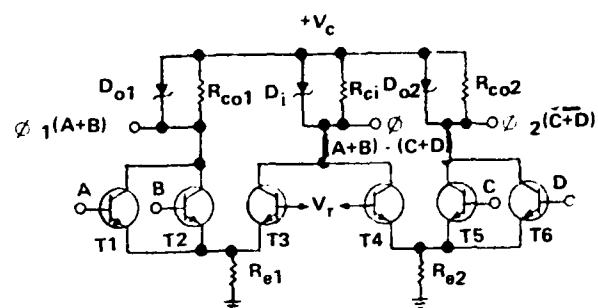


FIGURE 3.1-3 Schematic diagram of an in-phase collector dot circuit showing how the dot function can be formed by merging devices T3 and T4 into one device with a common collector bed, common base region, and two separate emitters. This arrangement saves space and does not require use of external or global wires to form the dot.

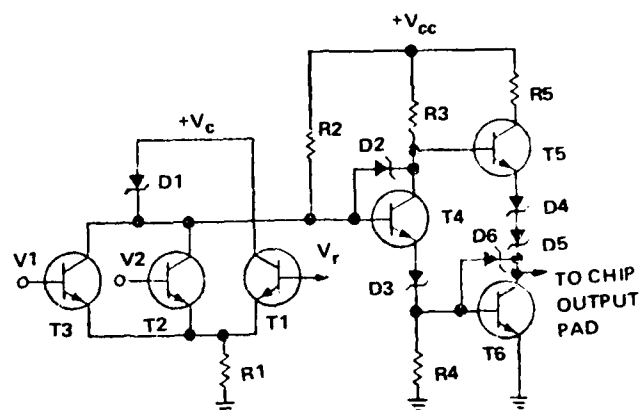


FIGURE 3.1-6 Schematic diagram of off-chip driver level shifting circuit.

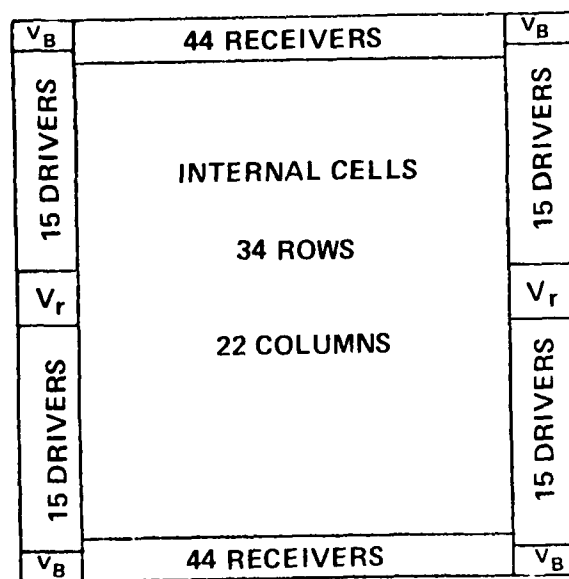


FIGURE 3.1-1 Physical arrangement of masterslice, showing locations of logic gates, level shifting circuits, and reference generator circuit.

degraded and impacting the circuits's noise margin. A typical gate delay for the low-power version is 1.5 ns loaded, at 0.85 mW, which gives a speed-power product of 1.28 pJ. The high-power version has twice the drive capability of the low-power gate and has a delay of 0.80 ns loaded, at 1.7mW. As shown in Figure 3.1-3, it is possible to take the standard OR/NOR circuit and perform either an in-phase or an out-of-phase collector dot which generates the AND function. Up to five-way in-phase collector dot is permitted, but the out-of-phase dot is limited to two because of possible saturation problems. Since there is a possibility of leakages when dotting, it is necessary to derate the fan-out capability of the circuit in order to prevent excessive degradation of the up level. It is also possible to construct polarity-hold latches to form registers. This is accomplished by feeding the output of an in-phase collector dot back into one of its own inputs, while providing the necessary clocking signals (see Figure 3.1-4).

The on-chip receiver circuit translates the incoming T^2L logic levels down to the on-chip levels detailed above. The schematic diagram is shown in Figure 3.1-5. The circuit is a common base configuration, with a Schottky barrier diode base-collector clamp (D2) to prevent saturation. Diode D1 is normally reverse-biased, and serves to limit the down-going excursion of negative reflection. The average nominal power dissipation of this circuit is 0.7 mW. The typical propagation delay of the receiver is about 1.0 ns. The maximum input voltage allowed is 4.4 volts. This means that any input tied to an open collector output must use a split resistor pair to limit the voltage.

The schematic diagram of the off-chip driver is shown in Figure 3.1-6. This circuit converts the on-chip logic levels defined earlier to standard T^2L levels and requires both the (V_c) and the high (V_{cc}) voltage power supplies. The circuit contains a two input predriver which allows the designer to perform logic functions in the driver. Deleting components T5, R5, and D4-D6 results in an open-collector driver. The average nominal power dissipation of the driver circuit is 5.7 mW. The delay through a driver circuit with several T^2L loads and 50.0 pF of capacitive loading is on the order 8-10 ns.

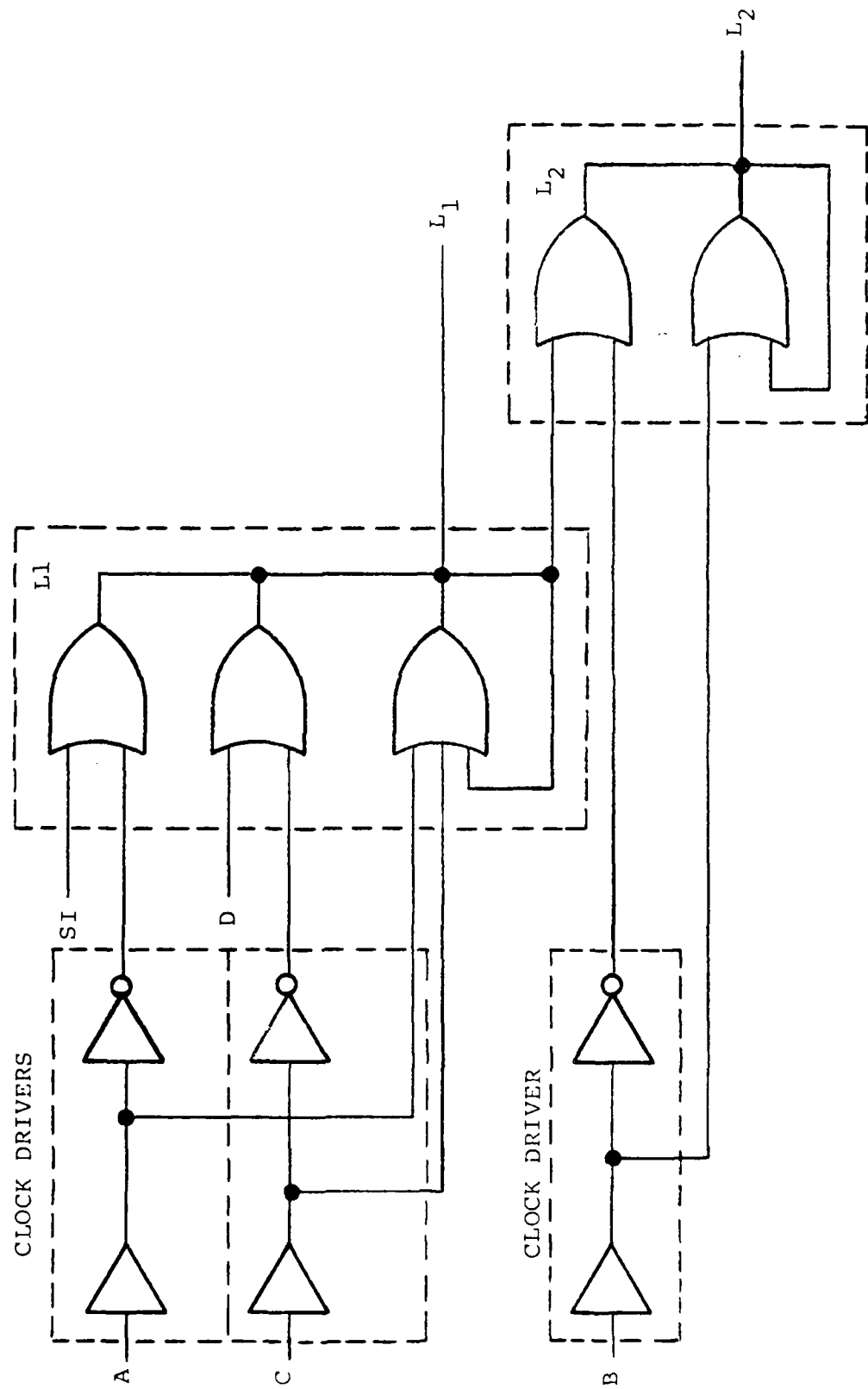


Figure 3.1-4 MS399 Shift Register Latch w/Clock Drivers

The terminal metallurgy of the chip consists of a 17 X 17 array of lead-tin solder balls. This array reduces voltage drops in the power distribution system and enhances chip cooling. The chip is mounted on a 28mm, single chip, metallized ceramic substrate with 116 pins. An array of solder balls identical to the one deposited on the chip is located at the center of the package. The chip is inverted, mounted in place, and the entire assembly is then put into an oven where the solder balls reflow. A metal cap is then added, and the module is sealed with an inert material. This flip-chip bonding technique is used because of its high I/O signal capability, and its compatibility with a high volume manufacturing environment. The use of multiple ground pins to minimize ground pin inductance reduces ground bounce when several off-chip drivers are simultaneously switched.

The entire manufacturing process takes place at IBM's manufacturing plant in East Fishkill, NY, beginning with crystal growth, continuing through mask design and fabrication, masterslice build and personalization, and ending with a completed and tested module. The chip is made using a bipolar process with a 2.0 μ -thick epitaxial layer. Starting with an 82mm P⁻ wafer, the normal steps consisting of subcollector and subisolation diffusion, epitaxial layer growth, collector reach-through diffusion, base and resistor diffusion, and emitter diffusion are performed.

Three levels of metal, separated by quartz insulation, are used for wiring. The masterslice is personalized on the first and the second levels of metal, as well as on the interconnecting or via level. Third level metal is used for I/O signal and power distribution and remains fixed. First level metal runs in the horizontal direction, and second level metal runs in the vertical direction. A lift-off process is utilized to produce very fine linewidths, and all three layers of metal are composed of an aluminum-copper alloy. A typical linewidth is 5.0 μ with a center-to-center spacing of 6.5 μ . For a part number with about 1200 circuits the total metal line-length on first and second level metal can be excess of 4.0 meters.

Converting the 74S481 design to the technology described resulted in the differences in pinout, packaging, power, electrical specifications and

increased signal switching speed described in Appendix A. Since the internal circuits are limited to OR/NOR logic gates with no analog comparator type circuits, the position input had to be implemented digitally. The design was also restricted by the logic macro's or books which were available at the time. Some high-powered versions of the books were not available, resulting in the use of duplicate or triplicate low-powered books to achieve the necessary fanout. The three-state driver books were also unavailable; therefore, all three-state signals were implemented as either open collector or active collector devices. The impact of these missing books will be discussed later.

3.1.2 LEVEL SENSITIVE SCAN DESIGN (LSSD) RULES

Any design using MS399 is required to adhere to Level Sensitive Scan Design (LSSD) Rules. Since the primary purpose of the contract is to compare LSSD to standard design techniques, this made MS399 an ideal candidate technology. In addition to the technology differences just described, conformance to LSSD Rules introduced some additional differences in the redesigned 74S481. A brief introduction to LSSD is necessary to understand these differences. The IBM LSSD method of design ensures race-free system operation as well as race-free testing. This design method is defined as follows:

Definition: A logic subsystem is level-sensitive if and only if the steady-state response to any allowed input state change is independent of the circuit and wire delays within the subsystem. Also, if an input state change involves the changing of more than one input signal, then the response must be independent of the order in which they change. Steady-state response is the final value of all logic gate outputs after all change activity has terminated.

A principal objective in establishing design constraints is to obtain logic subsystems that are insensitive to ac characteristics such as rise time, fall time, and minimum circuit delay. Consequently, the basic storage element should be a level-sensitive device that does not contain a hazard or

race condition. The polarity-hold latch meets these requirements, provided it is implemented properly.

The polarity-hold latch has two input signals (Figure 3.1.2-1). Its operation is as follows:

When $C = 0$, the latch cannot change state.

When $C = 1$, the internal state of the latch is set to the value of the excitation input D.

The testing problem can be greatly simplified if the latches are also capable of operating as shift registers (See Figure 3.1.2-2). In this manner test data and test responses may be inserted/obtained from within otherwise impossible logic structures. These level-sensitive scan registers are known as shift register latches or SRL's.

Observed benefits offered by this design discipline are:

- (a) System performance is not dependent on hard-to-control ac circuit parameters such as rise time, fall time, or minimum delay. It is dependent only on the longest path delay being less than some specified value. Clock skew and fast path problems are essentially eliminated for testing purposes.
- (b) Test generation and testing are simplified to the well understood method of combinational logic network testing. This allows reduction in test pattern volume and fault simulation time by using pattern subsumption (Section 3.9.4).
- (c) The ability to dynamically monitor the state of all internal storage elements is inherent in the design. This eliminates the need for special test points, simplifies manual debugging, and provides a standard interface for operator and maintenance consoles.

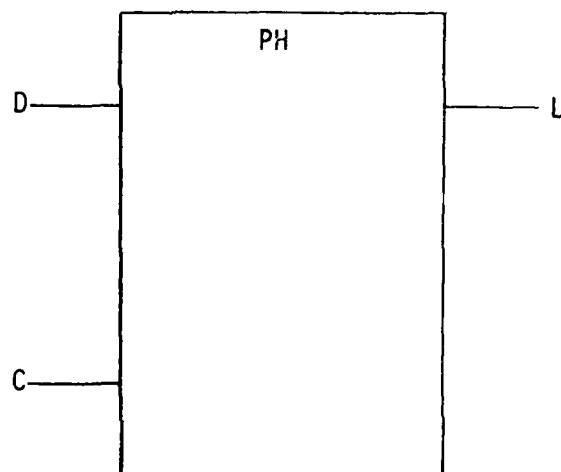


FIGURE 3.1.2-1 POLARITY HOLD LATCH



- d) The development and use of tools for design verification simulation and for checking is simplified.
- e) The insensitivity to timing problems and the modular design structure help reduce engineering changes and additional design passes.
- f) The level-sensitive design allows the use of a unit logic hardware simulator for development design without creating timing problems in the transition from unit logic to dense functional chips.
- g) The method used for testing chips and modules can also be used for diagnostic tests in the field. SRL's simplify design partitioning for more cost effective test generation.

The six basic design rules necessary for an LSSD design are listed in Table 3.1.2-3*. Most of the rules apply to the configurations and control of specific paths in the network such as the scan paths between SRLs, system data paths between SRLs, and the paths between clock primary inputs and the SRLs.

A rules checking program is used to verify conformance to these rules. The rules checker uses a logic simulation program to trace out the particular paths to which LSSD rules apply. This system has proven to be a fast and efficient method for allowing the designer to check his logic design for compliance with the design rules. It is virtually automatic, requires little or no new user knowledge, and provides diagnostic messages which permit the designer to quickly locate and correct problems. Many problem conditions not strictly related to LSSD design rules such as logic and timing errors are also detected.

* A Logic Design Structure For LSI Testability
 E.B. Eichelberg & T. W. Williams
 Design Automation Conference #14 (IEEE Cat 77 CH1216-1C)
 New Orleans, LA 6/22/77

TABLE 3.1.2-3

Summary of LSSD Design Rules

- Rule 1. All internal storage is implemented in hazard-free polarity hold latches (i.e., latches which are controlled by clock signals such that the data stored in the latches cannot be changed by other inputs when the clocks are "off").
- Rule 2. The latches are controlled by two or more non-overlapping clocks such that:
- a. A latch, X, may feed the data port of another latch, Y, if and only if the clock that sets the data into latch Y does not clock latch X.
 - b. A latch, X, may gate a clock C_i to produce a gated clock C'_{ig} which drives another latch, Y, if and only if clock C'_{ig} does not clock latch X, where C'_{ig} is any clock derived from C_i .
- Rule 3. It must be possible to identify a set of clock primary inputs from which the clock inputs to SRLs are controlled either through simple powering trees or through logic that is gated by SRLs and/or non-clock primary inputs. Given this structure, the following rules must hold:
- a. All clock inputs to all SRLs must be at their "off" states when all clock primary inputs are held in their "off" states.
 - b. The clock signal that appears at any clock input of an SRL must be controllable from one or more clock PIs such that it is possible to set the clock input of the SRL to an "on" state by turning any one of the corresponding clock PIs to its "on" state and also setting the required gating conditions from SRLs and/or non-clock PIs.
 - c. No clock can be ANDed with either the true or complement value of another clock.
- Rule 4. Clock primary inputs may not feed the data inputs to latches either directly or through combinational logic, but may only feed the clock input to the latches or primary outputs.
- Rule 5. All SRLs must be interconnected into one or more shift registers, each of which has an input, an output and shift clocks available at the terminals of the package.
- Rule 6. There must exist some primary input sensing condition (referred to as the scan state) such that:
- a. Each SRL or scan-out PO is a function of only the single preceding SRL or scan-in PI in its shift register during the shifting operation.

TABLE 3.1.2-3

Summary of LSSD Design Rules (continued)

- b. All clocks except the shift clocks are held "off" at the SRL inputs, and
- c. Any shift clock to an SRL may be turned "on" and "off" by changing the corresponding clock primary input for each clock.

Another important benefit is the generation of information used in the automatic test generation, testing, and repair processes. Additional types of checking can be added easily. Because other block types such as memory arrays may be represented as behavioral models the checking system may be extended to include very complex designs which contain a wide range of hardware. The system has been highly successful in practical use.

3.1.3 ALTERATIONS TO 74S481 DESIGN FOR LSSD CONFORMANCE

Figure 3.1.3-1 shows a rough second level diagram of the signal flow through a standard 74S481 design. All latches and flip-flops in this design operate off the same clock. Also note that no signal ever passes through more than two latches and/or flip-flops in going from a primary input to a primary output.

In order to conform with LSSD Rules and 74S481 timing the signal flow of the 74S481 must be altered as shown in Figure 3.1.3-2. To make the LSSD 74S481 function like the standard 74S481 the B and C_1 clocks should be generated equivalent to the C_K clock of the standard design. The C_2 clock needs to be generated as a nonoverlapping inversion of C_K . The A clock is used only for test purposes. The scan in pin (SI) could be shared with one of the other primary input pins; and the scan out pin (SO) could be any one of the register outputs since they are connected to primary input controllable muxes which connect directly to primary outputs. This means four pins, C_1 , C_2 , A, and B, are necessary to implement the clocks for the 74S481 in contrast to the one pin, C_K , for the standard design. This LSSD design has one additional disadvantage in that the A and B latches consist of two latches in the signal path instead of one, therefore, adversely impacting performance.

One of the changes made in the 74S481 design was the separation of the BI/O Bus into input and output busses. This alteration was made to increase the speed of the B Input Bus since MS399 technology restrictions did not allow three-state drivers for the output drivers (see Section 3.1.4.6 for explanation). An additional advantage accrued from separating the BI/O Bus;

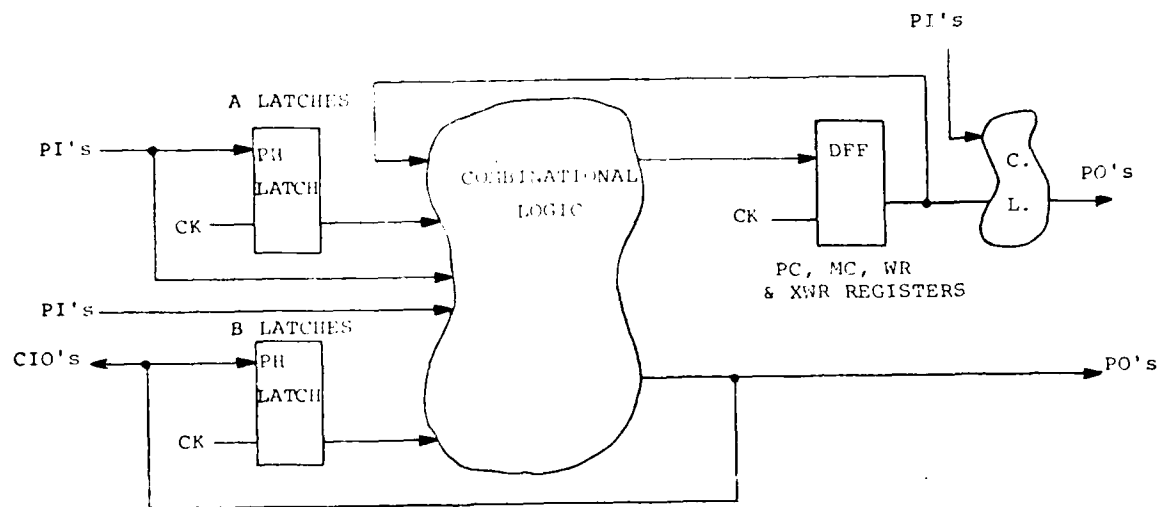


Figure 3.1.3-1 Standard 74S481 Design

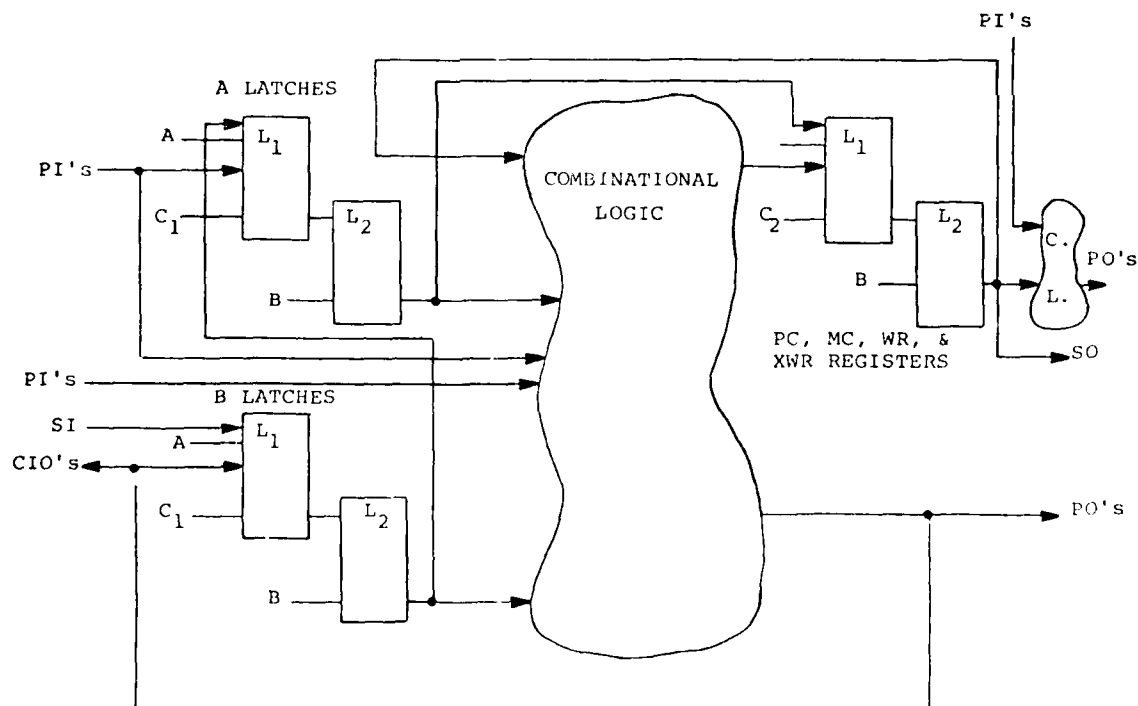


Figure 3.1.3-2 LSSD 74S481 Design

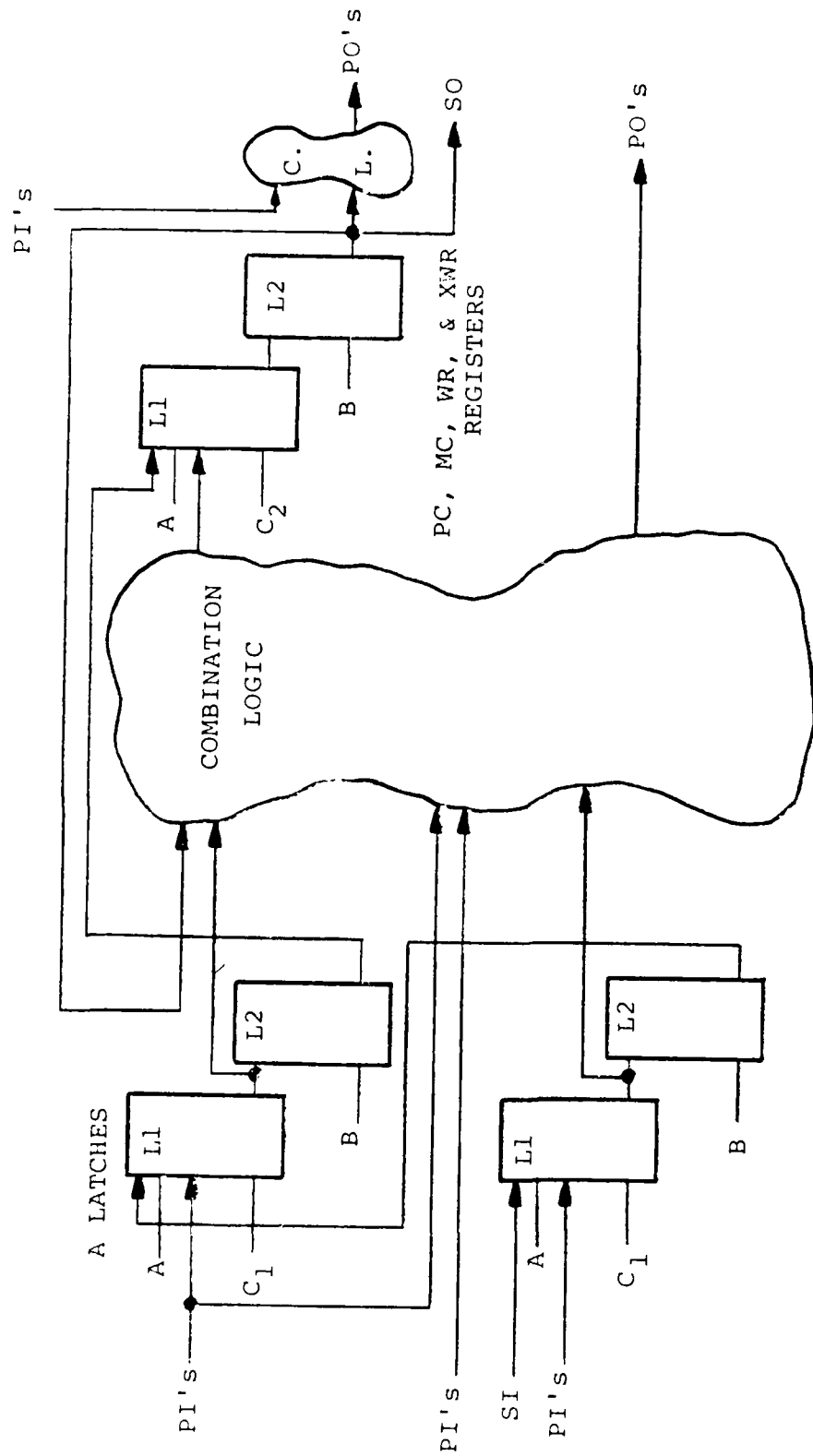


Figure 3.1.3-3 Modified LSSD 74S481 Design

the L1 outputs of the A and B latches can be used to feed the combinational logic. Rule 2a required that the L2 be in the data flow as long as the B Bus through B latch to B Bus feedback path existed since this latch would then feed its own data input. With this path eliminated, the B latch only feeds combinational logic and the address and data registers L1's. Rule 2a still requires that C_1 be different from C_2 , but the L1 output of the A and B latches can now be used. Figure 3.1.3-3 shows the signal flow diagram for this modified 74S481 design. The clocks are generated as previously defined, however, the L2 of the A and B latches no longer impacts performance.

3.1.3.1 MS399 Shift Register Latches

Figure 3.1.3-4 showed the logical representation of the basic MS399 SRL. Notice that the clock drivers provide an additional block of delay to the negative clock signal to guarantee proper data latch up on the falling edge of the clock. This delay in the clock line also guarantees a glitch from 0T1T0 on the rising edge of the clock. Also manual verification that the ratio of the capacitance on the plus clock line to the capacitance on the minus clock line is less than two is necessary to assure that skew caused by capacitance differences will not eliminate the delay between the two phases. These clock drivers can each fan out to 9 latches. Figure 3.1.3-4 contains the logical representation of the dual port SRL used in the stack design described later.

3.1.3.2 On Chip Clock Generation

In order to generate the two non-overlapping clocks described in Section 3.1.3 from the single system clock that appears on the 74S481 clock pin, some form of clock generation circuitry is needed. This circuitry was put on chip to save the external component count necessary to emulate the 74S481. A circuit which generates the proper clocks, yet does not violate LSSD Rules is shown in Figure 3.1.3-5. This circuit was implemented in the SCS481.

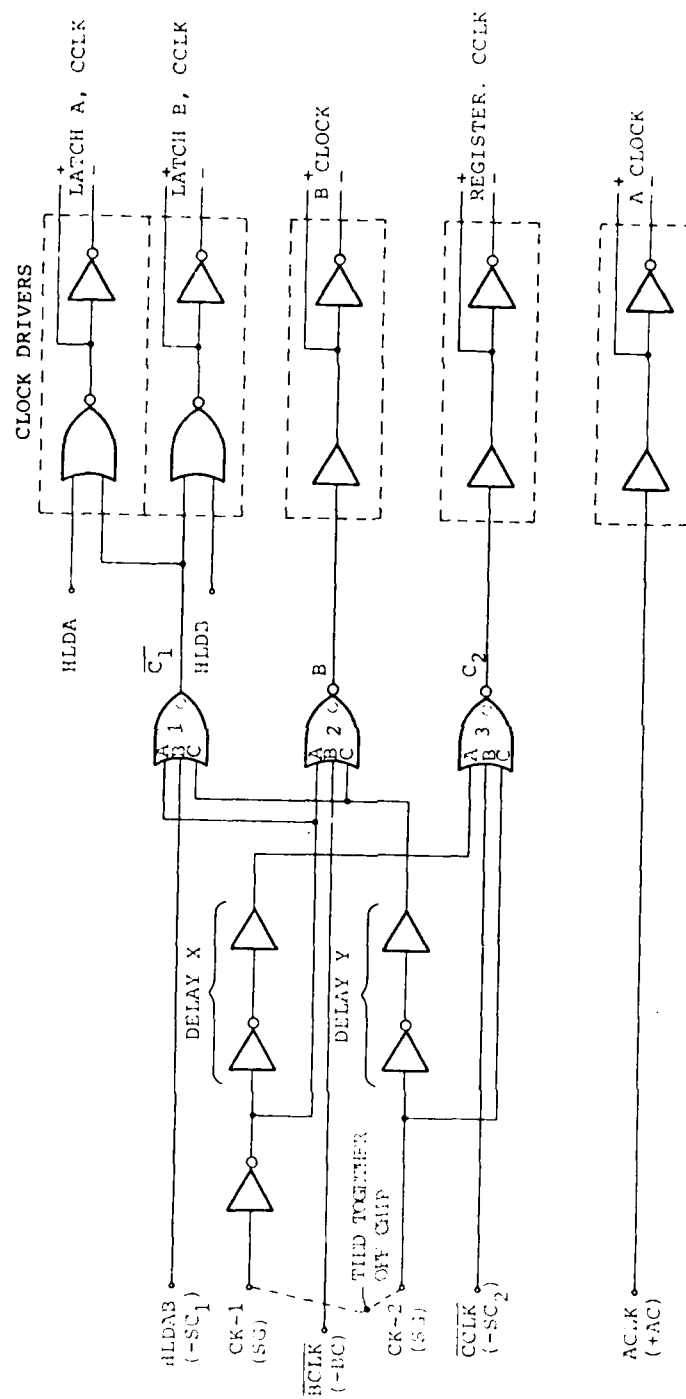


Figure 3.1.3-5 On Chip Clock Generator

For proper function of the clock generator the two lines CK-1 and CK-2 are tied together and used as the system clock input; all the other inputs are tied low. The inverter-buffer pairs Delay X and Delay Y prevent the various clocks from overlapping. The resulting waveforms are shown in Figure 3.1.3-6. Clocks C_1 and B are delayed equivalents of the system clock; and C_2 is a non-overlapping inversion of the system clock. These are the clocks which are required for proper operation as explained in Section 3.1.3. Delay Y determines the "dead time" between the falling C_2 and the rising B Clock which controls latch propagation delay (see Section 7.4). Both X and Y were selected as two block delays to achieve high performance but yet assure that clock skew and fast paths would not cause incorrect data to be loaded into the latches.

The other six inputs in the circuit in Figure 3.1.3-5 require some explanation. The inputs HLDA and HLDB are enhancements discussed in Section 3.1.4. ACLK is the A Clock required for scanning in LSSD. Rule 3 requires an explicit primary input for each clock which can always hold that clock off and is able to turn that clock on separately from other clocks. To meet this requirement HLDAB, \overline{BCLK} , and \overline{CCLK} are used. \overline{BCLK} is the explicit B Clock primary input; whereas HLDAB and \overline{CCLK} are the explicit C_1 and C_2 clock pins respectively. Note that the primary reason these pins exist is for test purposes; they serve no immediately obvious functional purpose.

The inputs CK-1 and CK-2 cannot be flagged as the B, C_1 , or C_2 clocks because this would violate Rule 3c since gates 1, 2, and 3 essentially 'AND' CK-1 and CK-2. This would also violate Rule 6b since whenever CK-1 and CK-2 are high, the A and B latch C Clocks and the system B Clock are on simultaneously. To satisfy Rule 6c CK-1 and CK-2 must be flagged as scan gates indicating they must be high to place the design in the scan state. This allows the B clock to be controlled from pin \overline{BCLK} only. CK-1 and CK-2 must be separate pins, otherwise input pins 1A, 1C, 2A, 2C, 3A, and 3C as well as the Delay X and Delay Y buffers become untestable because they would form a redundant DC path.

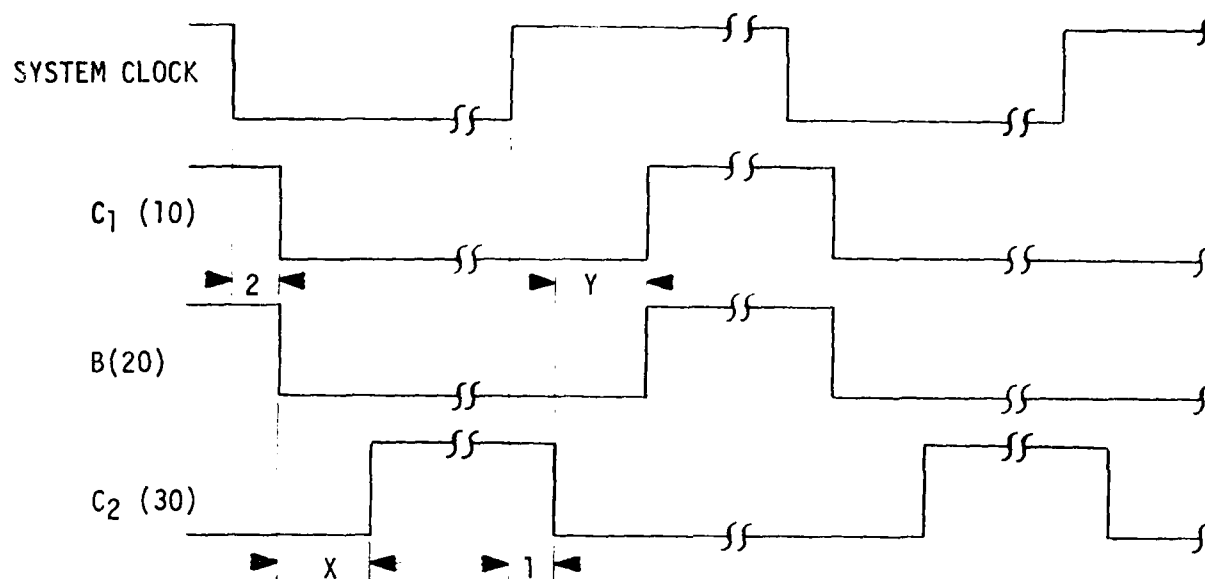
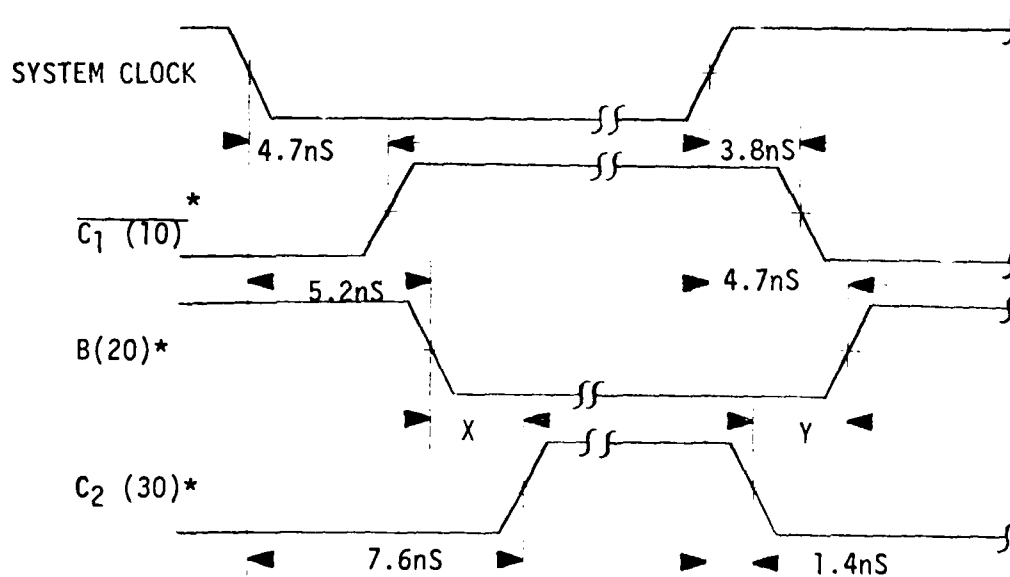


FIGURE 3.1.3-6 SCS481 CLOCK GENERATION TIMING (UNIT DELAY)



* - WIRED DELAY CALCULATIONS

FIGURE 3.1.3.7 ACTUAL SCS 481 CLOCK TIMING

Essentially, placing the clock generation circuitry on chip saved some external circuitry, but cost two additional I/O pins to conform to LSSD Rules and avoid redundancies. In general, on chip clock generation circuitry will cost additional pins in an LSSD design. If more than one LSSD device is used in a particular design, off chip clock generators are probably the wiser choice, especially since this allows the "dead times" to be under complete external control. By adjusting the "dead time", the designer can control the speed of the design and at least verify its DC function. In contrast, on chip clock generators are more or less "hardwired". If Delay X or Delay Y were selected at too small a value allowing fast path problems to develop, the other clock pins can be used to input signals generated off chip with larger "dead times". This provides a fail safe approach to verifying the DC function of the chip.

Figure 3.1.3-7 shows the "actual" clock generation timing for the SCS481 as determined by the delay calculator. Nominal delay calculation determines that Delay X and Delay Y should equal 3ns; i.e., two times the nominal delay of 1.5ns. "As wired" delay calculation shows X to be equal to 2.4ns and Y equal to 3.3ns. The majority of the difference is due to the fact that the B clock line drives three more clock drivers than the C₂ clock. This additional load slows gate 2 by an additional 0.3ns; therefore decreasing X and increasing Y. To improve the clock generator safety margin and increase latch speeds, X should be increased and Y decreased.

3.1.3.3 LSSD Rules Checking Program Violation Changes

The automatic LSSD rules checking program was run fairly early in the simulation phase of the design. Five feedback paths were identified as a violation of Rule 1. Three of these were discovered to be design errors which were immediately corrected. The other two feedback paths were the OP8 and OP9 common I/O lines.

The problem with OP8 was that the signal used to disable the OP8 output was being derived from signals which could be traced back to the OP8 input. Fortunately, the OP8 output disable signal was not dependent on OP8, so by decoding the opcodes differently this feedback was eliminated.

The OP9 output disable had similar problems; however, even when this had been corrected, a feedback path still existed. In a very simplified diagram, Figure 3.1.3-8 reveals the location of the feedback path. In Op Form I, OP9 controls the logical inversion of the B Bus. Since OP Form I performs an addition, OP9 determines the value of \overline{COUT} assuming all other inputs are constant. The \overline{COUT} signal feeds to gate #2 for use in the divide OP Forms XI and XII. Even though the op code decode logic prevents both gate #1 and gate #2 from being on simultaneously, the rules checking program is not smart enough to determine this. Therefore, the rules checking program identifies this as a potential feedback. OP9 has a similar problem for OP Form VII.

Figure 3.1.3-9 demonstrates the use of an SRL to break the "feedback" path. The C_3 and B_1 clock pins would functionally be tied high to make the latch transparent. This is the safest solution to this problem since it guarantees the "feedback" path is broken. The problem with this method is that it would use two additional I/O pins and 10 more half cells. The SRL also introduces about 5ns of delay in the OP9 output which is unacceptable since the OP9 output is the critical path in the multiply.

Another solution is illustrated in Figure 3.1.3-10. In this example, circuitry is added to generate a \overline{COUT} for division which is never dependent on OP9. This has the advantage of speeding up the divide operation, but uses approximately 40 additional half cells. Another disadvantage could be that even after implementing this solution, an even more obscure "feedback" path may be revealed which would require even more circuitry.

Figure 3.1.3-11 describes the solution actually implemented: breaking OP9 into an input and output pin. This solution has the advantage of using no additional cells and not affecting the speed of operation. The disadvantages are that it uses an additional pin and that the problem is merely passed on to the second level package. However, this was the simplest and most straightforward approach to use.

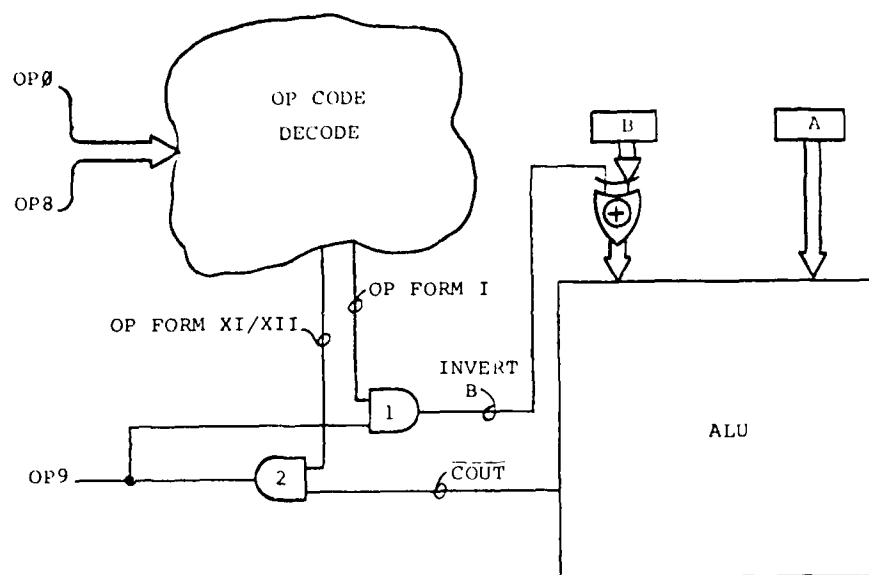


Figure 3.1.3-8 OP9 Feedback Problem

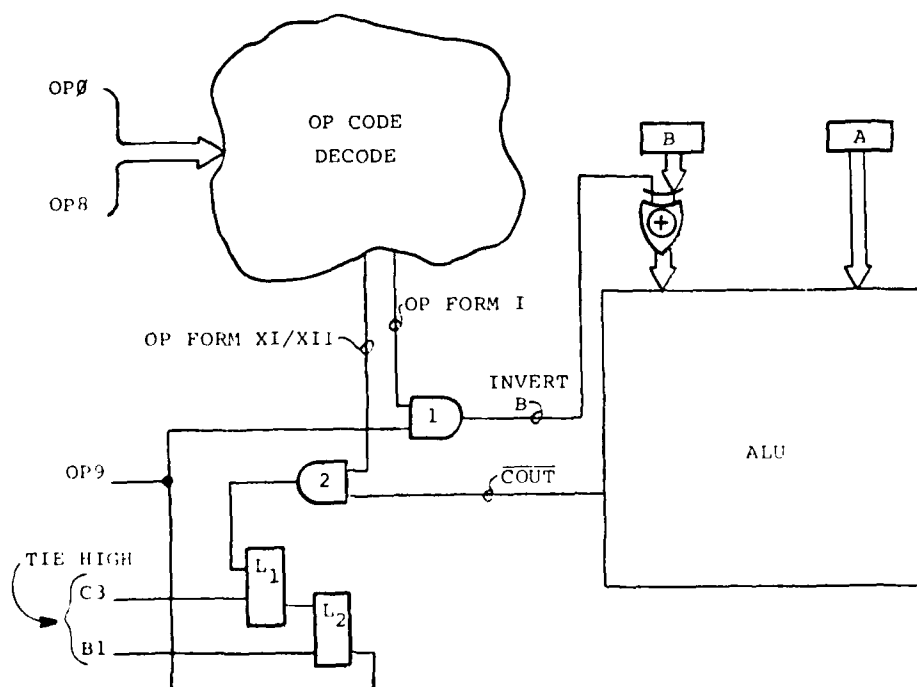


Figure 3.1.3-9 SRL Used to Break Feedback

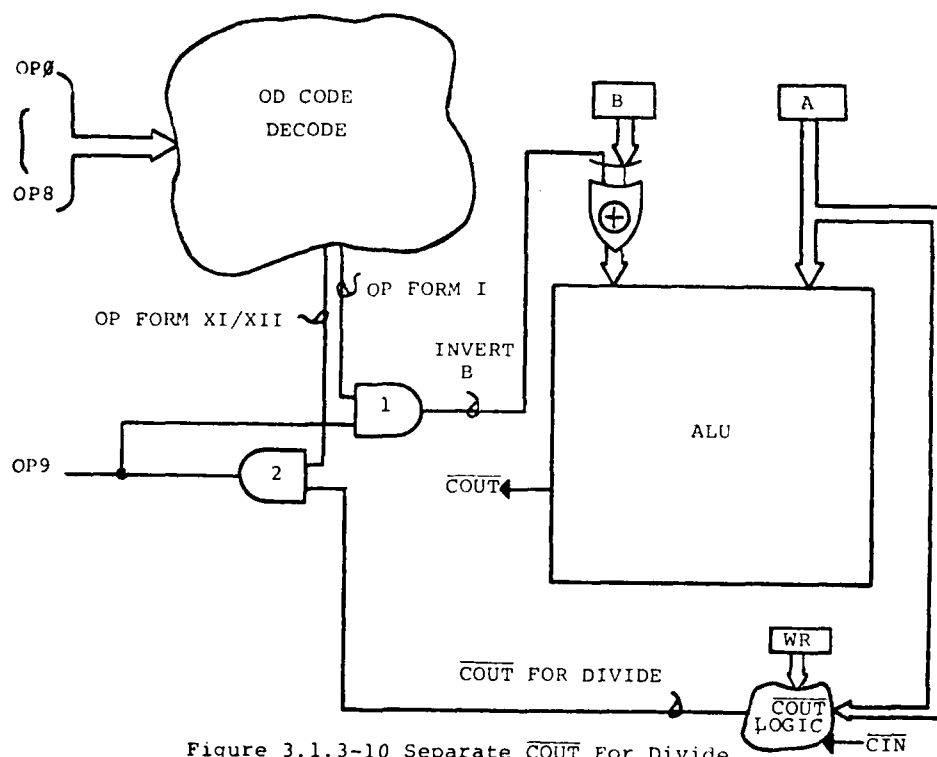


Figure 3.1.3-10 Separate \overline{COUT} For Divide

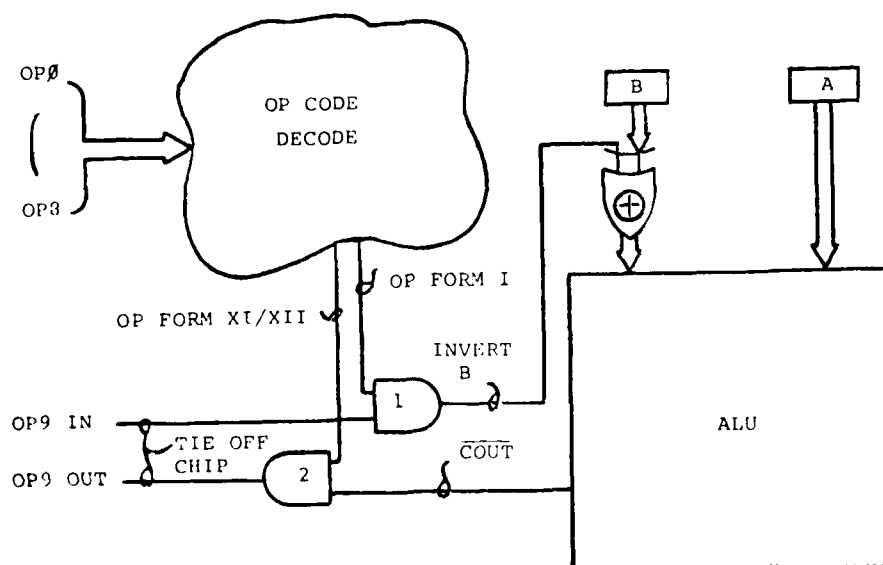


Figure 3.1.3-11 Separate OP9 Into Two Pins

Consultation with IBM LSSD experts concerning this "feedback" problem resulted in the conclusion that the original circuit in 3.1.3-8 was acceptable since the feedback path was mutually exclusive; i.e., both gate #1 and #2 cannot be on simultaneously. The problem is with the rules checking program which cannot detect this mutually exclusive situation. Suggestions have been made to correct this deficiency in the rules check program. For the present, though, recommendations are that any common I/O "feedback" paths be broken apart into two pins if extra I/O pins are available. This is because the test model generation program will also decide that this path has feedback and will attempt to cut the feedback for test generation. Since the test generators now think a non-LSSD feedback path exists, they may have difficulty generating tests in this area; therefore, these "feedback" paths should be broken when possible until the software can be enhanced.

3.1.3.4 Gated B-Clock Testability

As presently implemented, the scan function of the LSSD test generators always end with the application of the B clock. This practice results in L1 and L2 sharing the same data value. Testing any faults which can gate off the B clock require that the L1 and L2 be in opposite states. With the L1 and L2 in opposite states, one can gate off the B clock and pulse the B clock input. If no faults exist, L2 will remain in its present state otherwise it will switch values. Because of the current restriction that L1 and L2 must have the same value, the gated B clock signals was connected to a primary output to enhance (B-clock gate) testability.

3.1.3.5 LSSD Redesign Conclusions

Redesign of the 74S481 to conform to LSSD rules involved three basic areas: latch substitution, clock generation, and appeasement of the rules checking program. The net result is that three additional pins are absolutely necessary for an LSSD version of the 74S481, yielding an overhead of 6.5%. Nine additional pins were actually used in the design for LSSD. The scan in and scan out pins were broken out separately using two pins. The on chip clock generator required two more pins plus an output pin for the gated

B clock. The final extraneous LSSD pin was the OP9 output pin required to fool the rules check program. These six pins are not absolutely necessary for LSSD rules conformance.

The design data comparison section (3.1.7) concludes that 36 more gates (4.4%) are required for LSSD than would be required for a non-LSSD design in MS399. Eight of these gates are for the on chip clock generator, twenty-four are in the A and B latches for the unused L2 and scan capability, and a negative eighteen gates are due to the fact that an SRL requires one fewer gate than a DFF. The remaining twenty-two gates are for the various clock drivers. Only six of these gates are functionally necessary, two for each additional clock. If clock generation was performed off chip and the clock driver fanout could be increased, LSSD would cost only 12 additional gates or 1.5%. In any case, the LSSD costs in terms of I/O and gate count are relatively small.

3.1.4 ENHANCEMENTS

With the additional unused cells and I/O pins available in the SCS481, six enhancements were included in the design. These six enhancements are described below:

- (a) Bus from MC to A-Mux
- (b) Gate Clock to A, B Registers
- (c) 4X4 Stack for PC
- (d) Breakpoint Register
- (e) Dual OP9 outputs
- (f) B Bus Out separated from input

3.1.4.1 Bus From MC to A-Mux

In the present design of the 74S481, the memory counter (MC) cannot be used as a source in any of the OP Forms. To enhance the 74S481 design a bus was provided from the MC to the A-Mux. This can provide a path for computing relative address of MC data. A diagram of the altered A-Mux is shown in Figure 3.1.4-1. The LMC signal is active high and overrides any OP-Form A-Mux selection.

3.1.4.2 Gate Clock to A, B Registers

The clocks to the A and B registers were gated to permit arguments to be stored in the registers, allowing the A and B busses to be used for other purposes. Figure 3.1.3-5 shows the clock generation circuitry plus the two control lines added to create this enhancement. The HLDAB, HLDA, HLDB lines are all active high and control both the A and B register, just the A register, and just the B register respectively.

3.1.4.3 4X4 Stack for PC

A 4X4 stack was provided for the program counter. When the PC is used as a microsequencer, the stack permits efficient implementation of branch to subroutine and return from subroutines. A diagram of the stack and the PC are shown in Figure 3.1.4-2. The stack was made using the dual port SRL's shown in Figure 3.1.3-4.

When the PUSH control line is high and the $\overline{\text{ENSK}}$ line is low, a I transition on the clock causes the PC to be stored in Stack 0. The values in all the other stack registers is pushed up to the next higher stack register. The value in stack 3 is lost. When PUSH and $\overline{\text{ENSK}}$ are low, the I transition on the clock causes all the values in the stack registers to pop down to the next lower stack register. Stack 3 retains the same value on a pop.

The LSKPC signal is active high and overrides any OP Form controls to the PC. This signal loads Stack 0 into the PC on the I clock transition. If LSKPC is not high during a pop operation, the data in Stack 0 is lost. However, the PC can be loaded from Stack 0 without a pop by using LSKPC.

3.1.4.4 Breakpoint Register

A Breakpoint Register (BP) shown in Figure 3.1.4-3 was provided as a maintenance and debug tool. Two open collector outputs constantly compare the value of the 4-bits of the BP to the PC and MC, providing a low level when either one is equal. By dotting these outputs together an N-bit wide comparison can be made. The BP has its own input bus. Pulling signal $\overline{\text{LBR}}$ low allows the I clock transition to load the value on the BP Bus into the BP Register. The BP can also be read out of the BPTST port by applying low signals to the $\overline{\text{ENRO}}$ and TST lines.

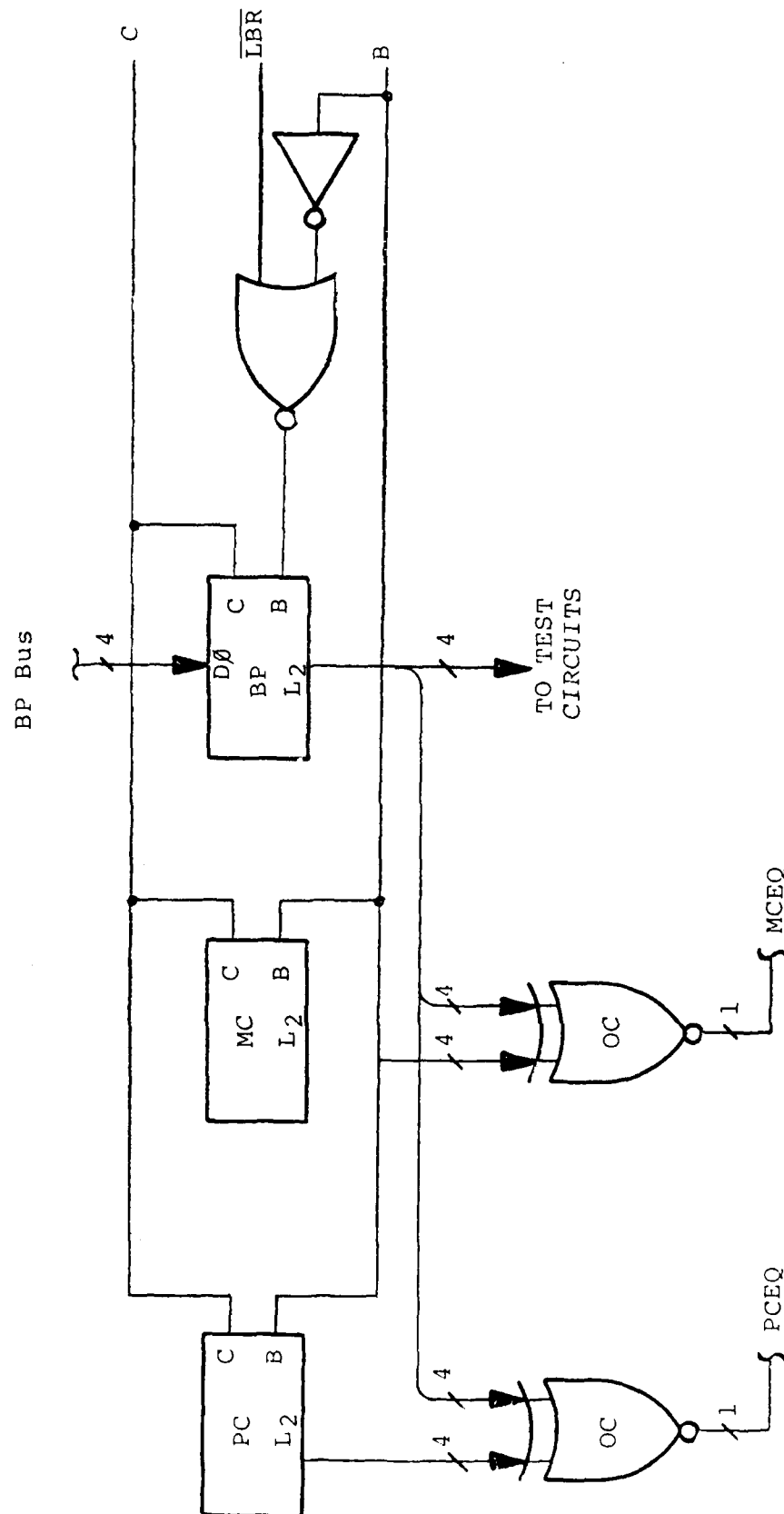


Figure 3.1.4-3 Breakpoint Register

3.1.4.5 Dual OP9 Outputs

As explained in Section 3.1.3.3, the OP9 pin was separated into an output and input pin. A duplicate output pin was added to enable a potential speed increase in OP9 propagation by splitting the OP9 load driven by the MSP. This additional drive capability is important because the OP9 propagation path is critical in the multiply instruction.

3.1.4.6 B Bus Out Separated from Input

Because three-state drivers were unavailable in MS399 (Section 3.1.1), open collector outputs would have been required to implement the B I/O Bus as it exists in the 74S481. Open collector drivers have the disadvantage of slowing the data propagation on the B Bus due to the pull-up resistors. These resistors would also create additional components on the test circuit boards, especially since split resistor pairs are necessary. Since the AP-101C does not use the B Bus Out and additional I/O pins were available, the B Bus Out was separated from the B Bus input. Addition of a three-state buffer, such as a 54/74126, to the B Bus Out port would easily allow emulation of the 74S481 B I/O Bus. This separation of the B Bus Output port also allows direct access to the Sum Bus.

3.1.5 TEST CIRCUITS

Since extra logic and extra I/O pins existed on the SCS481 after logic input of the 74S481 functions, four basic test circuits were implemented. These four circuits are described below:

- (a) Ring Oscillator
- (b) DOP Three-state Control
- (c) CEE0 Test Circuit
- (d) CEC8/CECO/CED8/CED7 Test Circuit.

3.1.5.1 Ring Oscillator

The ring oscillator was intended to provide a means of testing the basic logic speed of the technology, if desired. A logic diagram is shown in Figure 3.1.5-1.

3.1.5.2 DOP Three-state Control

The DOP Three-state Control circuit was pinned out as a test circuit to provide a signal to control a three-state buffer placed on the data out port. This circuit also provided a very simple logic function which could easily be tested. A logic diagram is shown in Figure 3.1.5-2.

3.1.5.3 CEE0 Test Circuit

The CEE0 Test Circuit was pinned out because it contained a newly defined logic function book CEE0. A logic diagram is shown in Figure 3.1.5-3.

3.1.5.4 CEC8/CEC0/CED8/CED7 Test Circuit

The CEC8/CEC0/CED8/CED7 Test Circuit was pinned out because it contained newly defined logic function books CEC8, CEC0, CED8, and CED7. A logic diagram is shown in Figure 3.1.5-4.

3.1.6 LOGIC DESIGN ERRORS

In the process of design, two logic errors were introduced which were not caught during design verification. The counter carry in (\overline{CCI}) for the least significant slice operates in an inverted fashion from the TI \overline{CCI} signal; i.e., a low causes a double increment and a high causes a single increment. All four shift bits, WRRT, WRLFT, XWRRT, and XWRLFT, are inverted for both input and output when compared to the TI signals. The circuitry necessary to correct these errors is described in section 3.8.4 and in Appendix A.

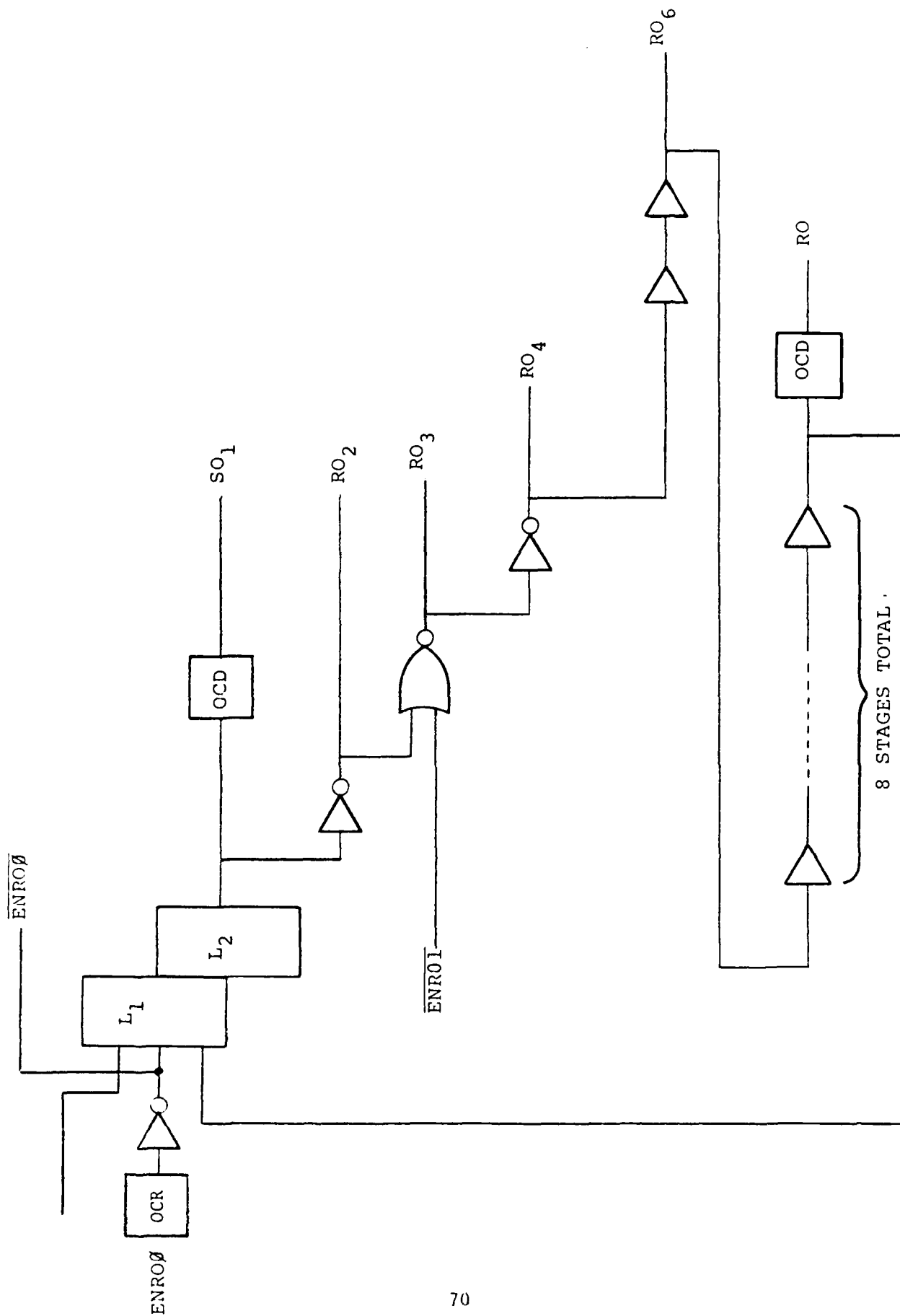


Figure 3.1.5-1 Ring Oscillator

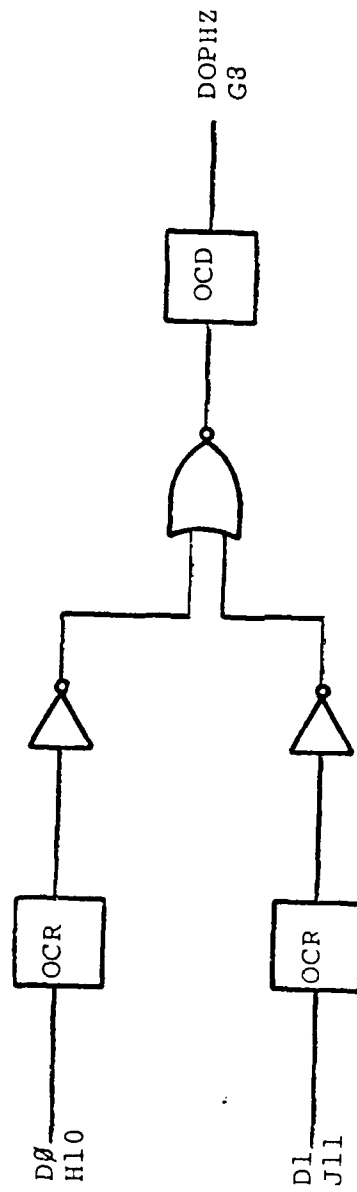


Figure 3.1.5-2 Three-State Control for DOP

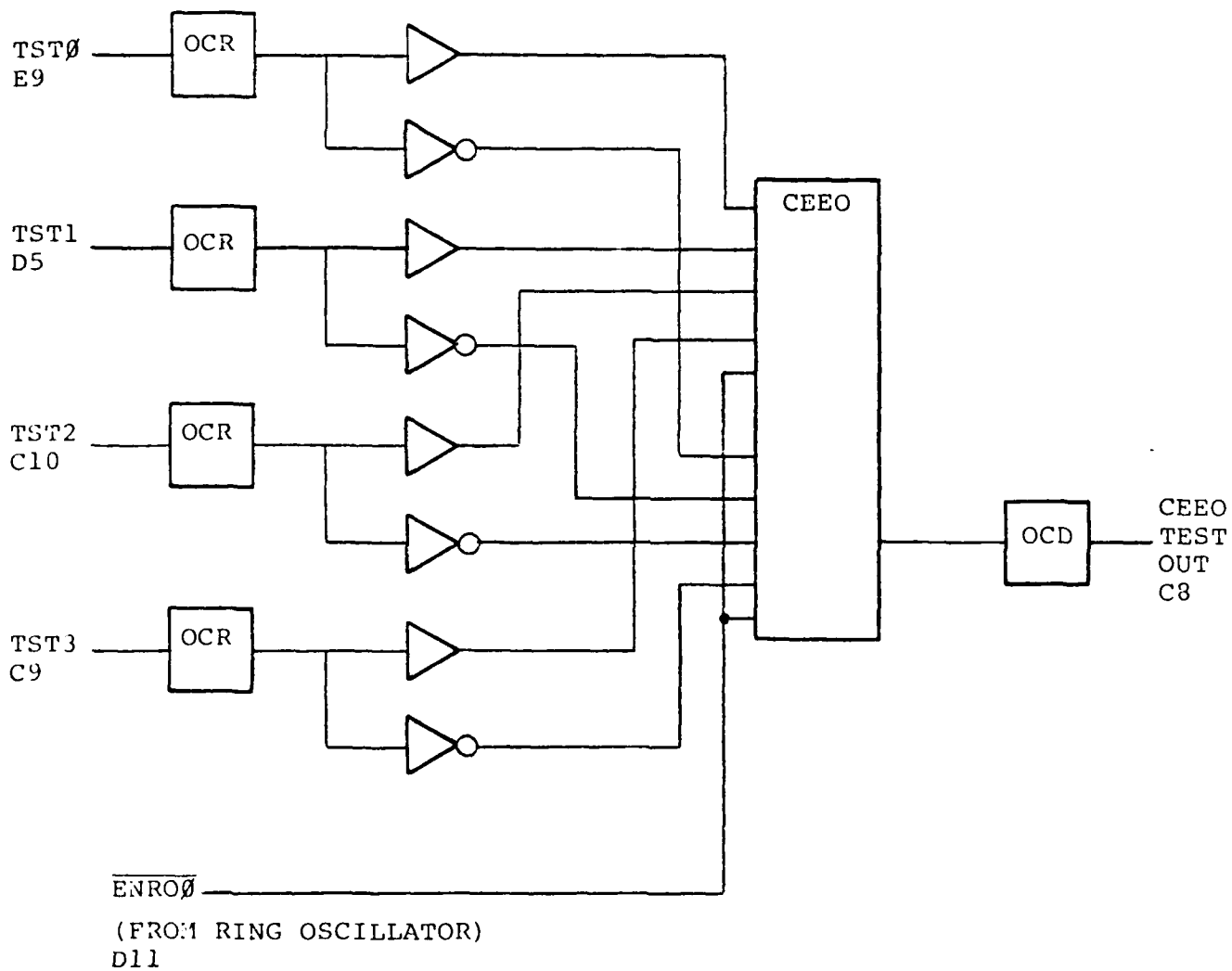
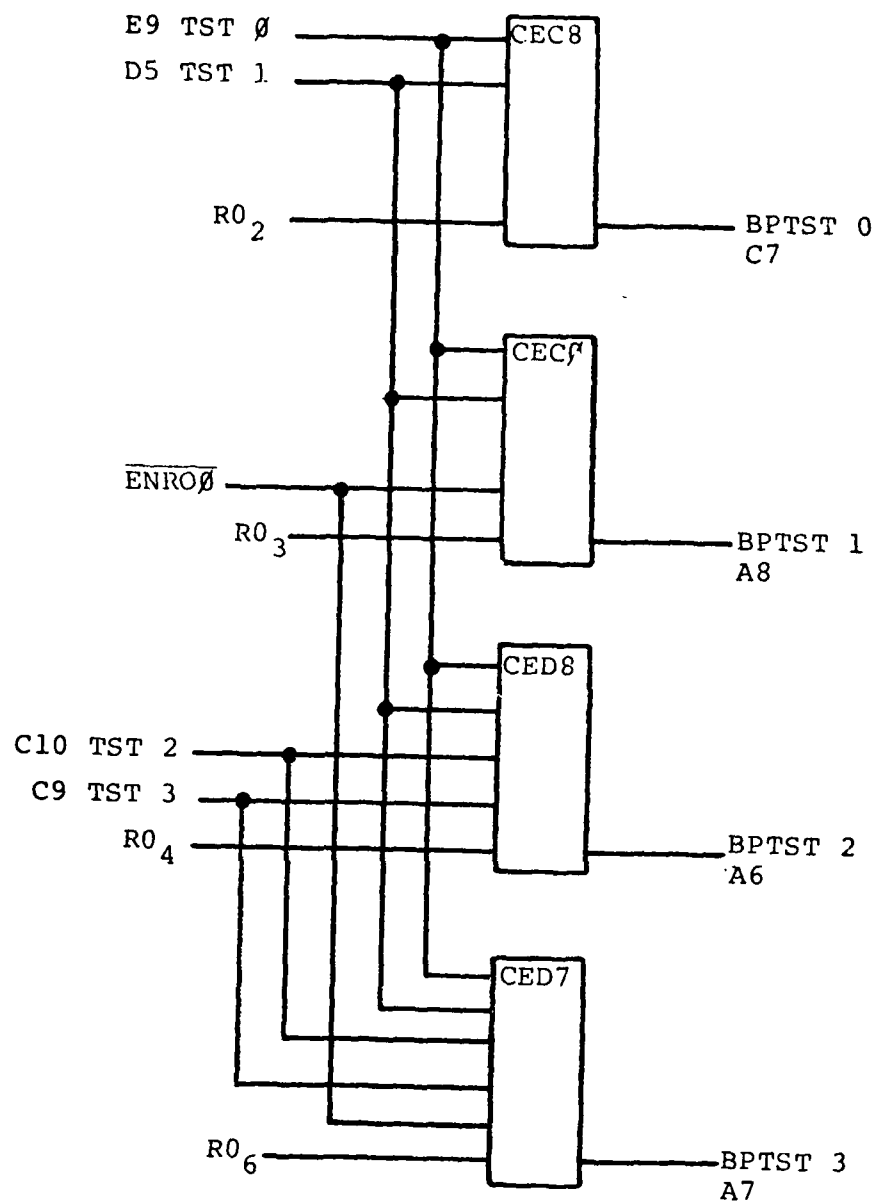


Figure 3.1.5-3 CEE0 Test Circuit



TEST SELECT		BP/TEST OUT			
0	1	0	1	2	3
0	0	BP ₀	BP ₁	BP ₂	BP ₃
0	1	BP ₃	BP ₀	BP ₁	BP ₂
1	0	BP ₁	BP ₂	BP ₃	BP ₀
1	1	R0 ₂	R0 ₃	R0 ₄	R0 ₆

Figure 3.1.5-4 CEC8/CEC0/CED8/CED7 Test Circuit

These errors were introduced very early in the logic design and remained because the logic designers assumed them to be correct. Since all simulation was written by the logic designers, all patterns applied tested these signals in the inverted state. Since discrepancies existed between the TI logic diagrams, the specifications in The Bipolar Microcomputer Components Data Book, and the real 74S481, and since design responsibility changed in the middle of logic entry, the introduction of logic inversion errors is easily understood. Nevertheless, these errors do not inhibit the use of SCS481's with other SCS481's; they work very well together. The only problem comes when one must replace TI 74S481 with an SCS481. Then the correction circuits must be added.

3.1.7 DESIGN DATA COMPARISON

A comparison of gate and pin count between the 74S481 and the SCS481 broken down by function is shown in Table 3.1.7-1. The gate count shown for the 74S481 is an approximation based on counting gates in the TI Logic diagram. The SCS481 gate count is based on the number of half cells plus any drivers or receivers used. The first four columns under each device indicate pin counts. The columns are for input pins, output pins, common I/O and total pins. The total column is reached by adding input and output pins and then subtracting common I/O.

The 74S481 functional subtotal show 46 total I/O for the 74S481 and 59 total I/O for the SCS481. The difference of thirteen pins can be broken down to one pin for digital position input, three pins for breaking the B Bus apart, and nine pins for LSSD considerations even though only three pins are required (see 3.1.3.5). The SCS481 adds an additional fourteen pins for enhancements, thirteen pins for test circuits, and twelve pins for power resulting in the total of 98 pins.

Table 3.1.7-2 shows a breakdown of the 74S481 functional subtotal. This table subtracts out those gates not directly comparable between the two designs. The embedded gates number indicates the half cells used which do not actually have wired circuits. At present, EDS rounds macro's which use an odd number of half cells to the next even number. Therefore, these unused cells are subtracted out.

The gate representation logic diagrams for the T1 74S481 did not explicitly indicate any drivers or receivers which were used in the design. For this reason, the 74S481 gate count does not take in to account any drivers and receivers. In order to make an equivalent comparison of internal gates required for both 74S481 and the SCS481, the drivers and receivers are subtracted from the SCS481 gate count.

As mentioned towards the end of Section 3.1.1, some high power book versions were unavailable at the time of the logic design. This required low power gates to be paralleled to achieve necessary fanout. With the high power books presently available, sixty-three half cells could be saved. An additional thirty-eight high powered gates are paralleled for fanout purposes; but no plans for a very high powered gate which could handle this fanout presently exist for MS399. These thirty-eight gates could be eliminated in a custom design; but for this calculation they are retained.

Figure 2.5-3 detailed the NOR logic model of a DFF. If this DFF were to be implemented in MS399, it would take up 6 half cells. The SRL shown in Figure 3.1-4 takes up only 5 half cells, not counting clock drivers. These facts were taken into account when figuring the LSSD gate count of 36. Eight of these gates are for the clock generator, twenty-four are the unused L2 and scan input of the L1 in the A and B latches, twenty-two are clock drivers, and a negative eighteen are due to the fact that an SRL requires one less half cell than a DFF. As pointed out in Section 3.1.3.5, only twelve of the gates would be necessary in a custom design.

Table 3.1.7-1. TI 74S481 To IBM SCS481 Comparison

Function	Texas Instruments 74S481					IBM SCS481				
	I P	O P	C O	Total	Gates	I P	O P	C O	Total	Gates
INSTRUCTION DECODE	10	2	2	10	128	10	2	1	11	247
ARITH-LOGIC UNIT	10	8	4	14	246	9	8	-	17	349
ADDRESS RESISTERS	4	5	-	9	129	4	5	-	9	179
DATA REGISTERS	7	8	4	11	153	7	8	4	11	217
MACRO LOGIC	-	-	-	-	60	-	-	-	-	77
CLOCK LOGIC	1	-	-	1	2	7	2	-	9	42
POSITION	1	-	-	1	5	2	-	-	2	9
SUBTOTAL	33	23	10	46	723	39	25	5	59	1120
OP9 ENHANCEMENT	-	-	-	-	-	-	1	-	1	13
STACK	-	-	-	-	-	3	-	-	3	123
BREAKPOINT	-	-	-	-	-	5	2	-	7	57
MC THRU A MUX	-	-	-	-	-	1	-	-	1	11
HOLD A,B REGISTERS	-	-	-	-	-	2	-	-	2	2
SUBTOTAL	33	23	10	46	723	50	28	5	73	1326
TEST CIRCUITS	-	-	-	-	-	6	7	-	13	72
TOTAL LOGIC	33	23	10	46/46	723	56	35	5	86/94	1398
POWER PINS	2	-	-	2	-	12	-	-	12	-
TOTAL	35	23	10	48/48	723	68	35	5	98/106	1398
POWER DISSIPATION	1.9 WATTS NOMINAL					1.3 WATTS NOMINAL				

Table 3.1.7-2. Base Function - Technology Efficiency Comparison

	Texas Instruments 74S481 Gates	IBM SCS481 Gates	
		W/O Enhance	W/ Enhance
TOTAL	723	1120	1398
LESS: EMBEDDED GATES	-	(55)	(65)
LESS: DRIVERS/RCVRS	-	(64)	(91)
LESS: HI-POWER DUP.	-	(63) *	(75)
LESS: LSSD LOGIC	-	(36) @	(13)
LESS: REDUNDANCIES	-	(91)	(91)
NET TOTAL	723	811	1063

* An additional 38 gates are paralleled for FANOUT purposes.

@ 16 of these gates are paralleled for FANOUT leaving only 20 as functionally necessary

Logical redundancies which were introduced into the logic could be removed by a redesign. Removal of these redundancies would save a total of 91 gates. After all these numbers have been subtracted from the 74S481 functional subtotal of 1120, a net total of 811 gates remains. This number represents the number of internal half cells that would be used if a non-LSSD redesign of the 74S481 in MS399 was realized.

3.1.7.1 Masterslice vs. Custom Implementation

In essence the difference of 88 gates between the 723 gates for the 74S481 and 811 for the SCS481 represents the cost of implementing this design in a masterslice instead of a custom approach. This amounts to a 12.2% increase in gates used. The majority of the additional cost was in the instruction decode logic. The 74S481 uses a PLA to decode the OP Codes, and, therefore more efficiently generated the necessary control signals. The primary cause for this difference is the fact that gates with more than three inputs use additional half cells and many of the TI PLA terms required seven or more terms. This cost is partially offset by the fact that the true and complement values are both available from each MS399 gate, greatly saving on inverter use.

An additional difference between the custom and masterslice approach was the fanout limitation of the gates. Twenty-two (38 minus 16 for LSSD) high-powered gates had to be paralleled in the combinational section of the logic. If the fanout limit of the gates could be increased with a very high powered version, these twenty-one cells could be eliminated. This would reduce the masterslice cost of 9.1%.

3.1.7.2 LSSD Implementation Impacts

The logic which performs the 74S481 function in the SCS481 required thirty-six additional gates and five additional pins to conform to LSSD rules. This represents a 4.4% increase in gates and a 10.9% increase in pins. The on chip clock generator contributes 1% to the gate count (8 gates) and 4.3% to the pin count (2 pins). The remaining 28 gates are divided

AD-A135 899

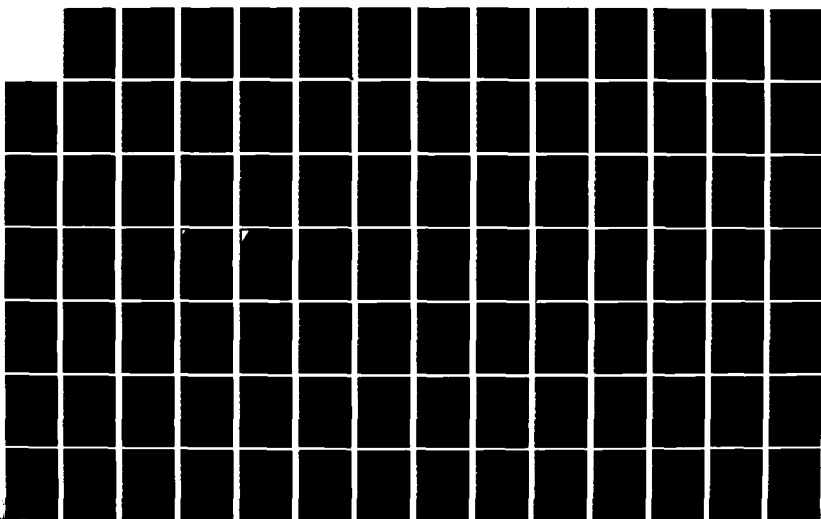
LSI (LARGE SCALE INTEGRATED) DESIGN FOR TESTABILITY
FINAL REPORT OF DESIGN (U) IBM FEDERAL SYSTEMS DIV
MANASSAS VA R D GROVES ET AL. NOV 83 AFWAL-TR-81-1068
F33615-79-C-1729

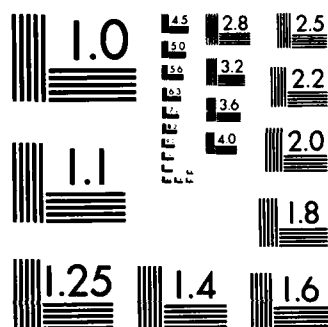
2/4

UNCLASSIFIED

F/G 5/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963 A

between the clock drivers (22 or 2.7%), the L2's and scan port of the L1's in the A and B latches (24 or 2.4%) and the decrease due to SRL vs. DFF size (-18 or -2.2%). The bare minimum requirement for an LSSD version of a 74S481 in MS399 is 28 additional gates and three extra pins (Section 3.1.3.5) provided off chip clock generation is used. This results in a 3.5% increase in gates and a 6.5% increase in pins.

If the A and B latches in the 74S481 had originally been DFF's instead of polarity hold latches, the minimum LSSD gate cost would have been only -0.5%. In general, mapping an edge-triggered DFF design into an MS399 LSSD design only requires a few additional clock pins. This is because MS399 implements the SRL in one less half cell than the DFF.

In fact, the MS399 dual port SRL in Figure 3.1.3-4 actually occupies two less half cell than the equivalent dual port DFF. Therefore, the LSSD gate cost for the stack is really a negative 32 gates. This accounts for the drop to thirteen in the LSSD gate count for the SCS481 with enhancements. The enhancements required no additional pins for LSSD conformance.

The minimum non-LSSD implementation of the enhanced SCS481 would use 1063 half cells and 77 pins (86 minus 9 for LSSD). The minimum LSSD implementation of the enhanced SCS481 would require 1068 half cells and 80 pins (assuming off chip clock generation). The resulting percentage increase is 0.5% for half cells and 3.9% for pins. The SCS481 implementation with on chip clock generation would require 1076 half cells and 82 pins. This is a 1.2% half cell count increase and an 6.5% pin count increase. The net result is that in any of these situations, the LSSD design cost is relatively insignificant.

If an LSSD 74S481 had been designed as a custom SCS chip with PLA, Section 3.1.3.5 points out that only 12 additional gates are required for LSSD. This represents only 1.7% increase above the 723 gates presently required for the non-LSSD 74S481. The LSSD cost for this custom design appears to be less than that for the masterslice design. However, even though a gate count comparison provides a good cost estimate in a masterslice design, an area comparison would provide a more meaningful comparison for a custom design. Unfortunately, insufficient information is available to properly estimate the increased area which would result from LSSD for this design; so a gate count must suffice. All of these LSSD and masterslice cost comparisons are summarized in Table 3.1.7.2-1.

In general, the gate penalty for LSSD is dependent upon the technology used and the kind and number of memory element being replaced. This fact is explained in detail in section 7.1. In most technologies, no significant gate count difference exist between a DFF implementation and the SRL implementation. For pin costs, the minimum number of additional pins that must be added is two, though most designs have at least three. The only time significant penalties are paid for LSSD is when a single polarity hold latch must be replaced with an SRL or when clocks are to be generated on the chip. Otherwise the additional internal circuits and I/O pins required for LSSD are primarily in the clock pins and their distribution system.

TABLE 3.1.7.2-1 DESIGN DATA COMPARISON

	GATES	SIGNAL PINS
1. 74S481	723 (est)	46
2. NON-LSSD SCS481 (BASE FUNCTION)	811 .	46 @
3. LSSD SCS481 (BASE FUNCTION)	839 *	49 *@
4. LSSD SCS481 (BASE FUNCTION (WITH ON CHIP CLOCK GENERATION)	847	51 @
5. NON-LSSD SCS481 AND ENHANCEMENTS	1063	77
6. LSSD SCS481 AND ENHANCEMENTS	1068 *	80 *
7. LSSD SCS481 AND ENHANCEMENTS WITH ON CHIP CLOCK GENERATION)	1076	82
8. LSSD CUSTOM 74S481	735 (est.)	49
	GATES	PINS
MASTERSLICE (1) vs CUSTOM DESIGN (2) COST	12.2%	0%
WITHOUT POWER DUPLICATION	9.1%	0%
LSSD vs. NON-LSSD DESIGN COST		
74S481 FUNCTION ON CHIP CLOCK GENERATION (2vs4)	4.4%	10.9%
OFF CHIP CLOCK GENERATION (2vs3)	3.5%	6.5%
SCS481 FUNCTION ON CHIP CLOCK GENERATION (5vs7)	1.2%	6.5%
OFFCHIP CLOCK GENERATION (5vs6)	0.5%	3.9%
74S481 CUSTOM DESIGN ESTIMATE (1vs8)	1.7%	6.5%

* These gate and pin counts assume the minimum LSSD requirements including off chip clock generation.

@ These pin counts do not include the additional pin for the digital postion input.

† This gate count includes 21 high powered gates paralleled for FANOUT.

3.2 LOGIC ENTRY TO DATA BASE (LSSD)

Once the logic has been designed, it must be entered into the Engineering Design System (EDS). Figure 3.2-1 gives an overview of the logic entry and design verification methodology. The following is a description of the data base as well as the various languages, programs, and system outputs used by the designer to aid in logic entry.

3.2.1 COLLECTION OF INDEXABLE DATA (CID)

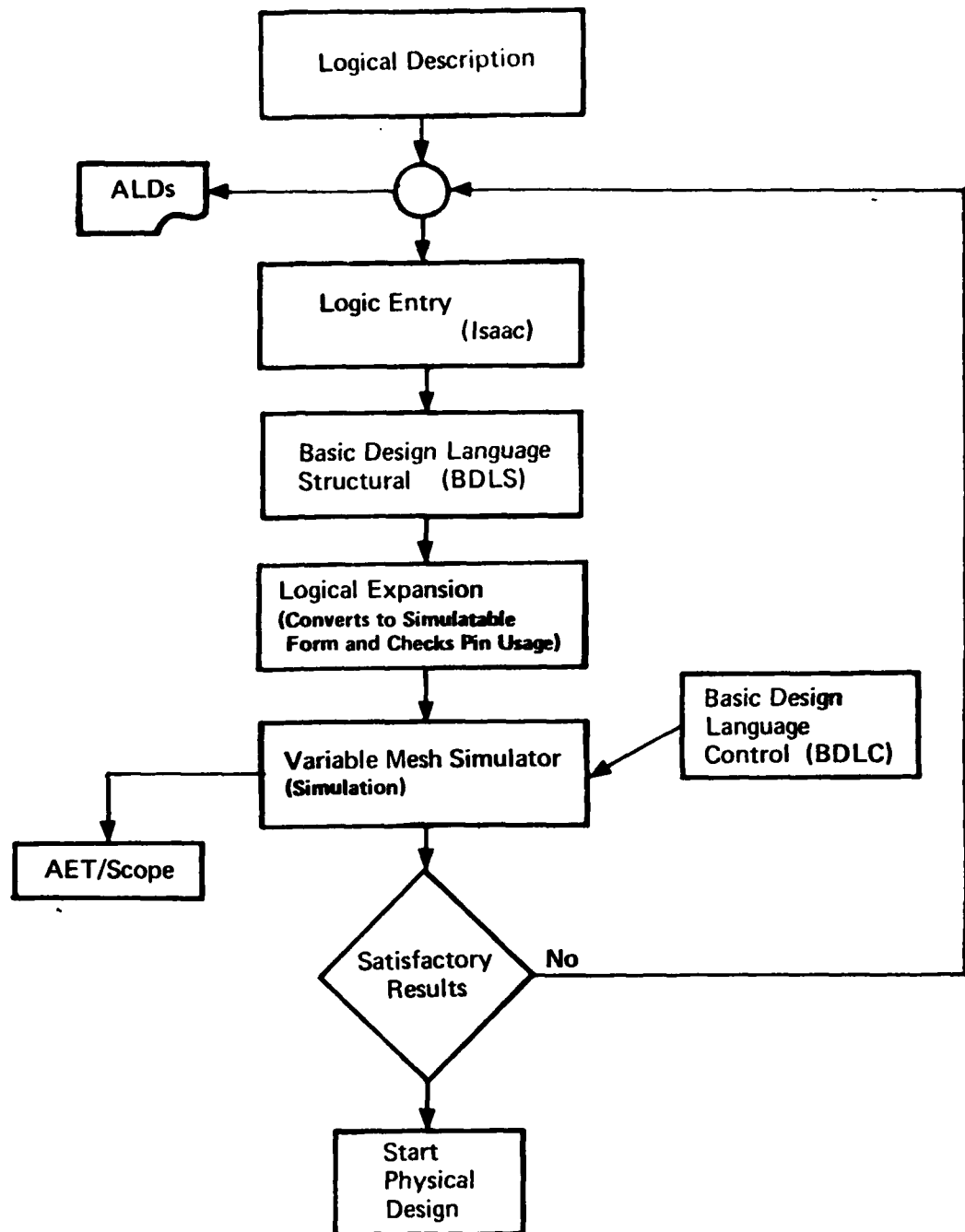
The design file is where the EDS user's design data resides. This includes not only the basic logic and physical data, but also the various levels of the design and supporting ALDs for each level of the design. The design data requires a unique structure to be useful and manageable. For this reason, EDS has developed a design file called a Collection of Indexable Data (CID).

The CID can be thought of as a single file into which the design data is placed. Actually, the CID comprises four separate data sets: Control data, index data, logical design data, and physical design data:

The control data set contains information and parameters that apply to the overall CID. This data is used by EDS to identify the CID and perform such functions as recovery from destructive failure or restoration to a previous design level.

The second section, the index data, is the key to locating specific types of information in the logical and physical design data areas. The index area keeps track of actual locations of the design data.

The last two sections of the CID--the logical (ALD) data set and the physical (PD) data set--are, from the user's point of view, the most valuable portion of the CID. These sections contain the actual design data, which are maintained and located by the control and index data areas. All EDS applications such as logic update, physical design, and simulation use



LOGIC ENTRY/SIMULATION

Figure 3.2-1

the CID as a data base. The CID--in conjunction with EDS rules--contains all physical and logical data for a manufactured unit. The CID provides version and engineering change level capabilities that allow many variations and stages of a design to be stored in and retrieved from a single data base.

3.2.2 LOGIC BOOKS/MACROS

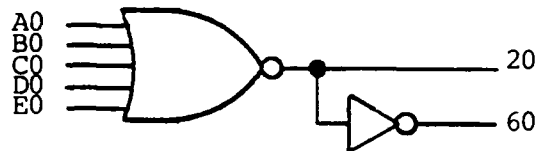
Books are predesigned connections of devices that form physical/logic circuits in the predefined masterslice cells. Each book is designed to have variable input and output "pins" for global logic wiring. Each book is automatically connected to appropriate power supplies, and internal interconnections are predefined. Use of these books greatly simplifies the logic design of an integrated circuit. Rules representing the physical implementation of the book and the logical model, including calculated delay equations, of the book are prewritten and available to the logic designer. Those MS399 books which were used in the design of the SCS481 are listed with their logical representation in Figure 3.2.2-1.

3.2.3 BASIC DESIGN LANGUAGE FOR STRUCTURE (BDL/S)

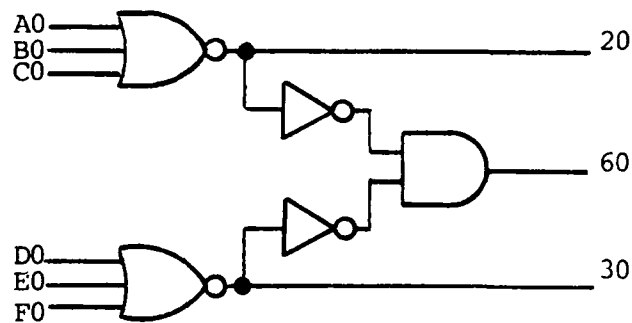
Design data is entered into the CID using equation-like design language statements called BDL/S (Basic Design Language for Structure). BDL/S is a high-level language; that is, many pieces of design data can be coded in an individual BDL/S statement. For example, all information associated with a conventional ALD logic block may be contained in one BDL/S statement.

Each BDL/S statement describes one function. This function may be operational, such as a logic book storage device. Or it may be a basic logic function (such as AND, OR). The EDS user may write BDL/S statements to describe high-level functions for the early modeling activity, but the greatest part of BDL/S input describes basic logic design.

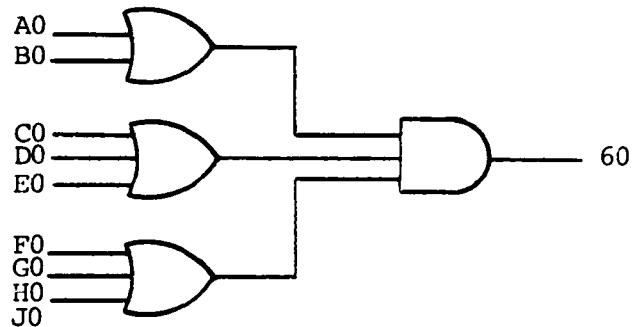
CEB0, CEB2, CEB9, CEC1, AND CEC3
1-5 INPUT OR/NOR GATE



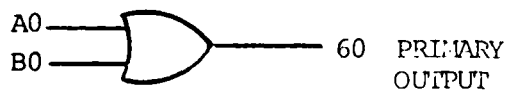
CEB4
2-6 INPUT OR - AND GATES



CEB5
3-9 INPUT OR - AND GATES



CEC5 (OPEN COLLECTOR, CEC7 (ACTIVE)
OFF CHIP DRIVERS



CEC6, CEE1
ON CHIP RECEIVERS

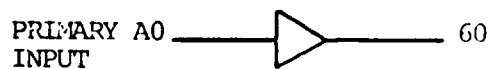
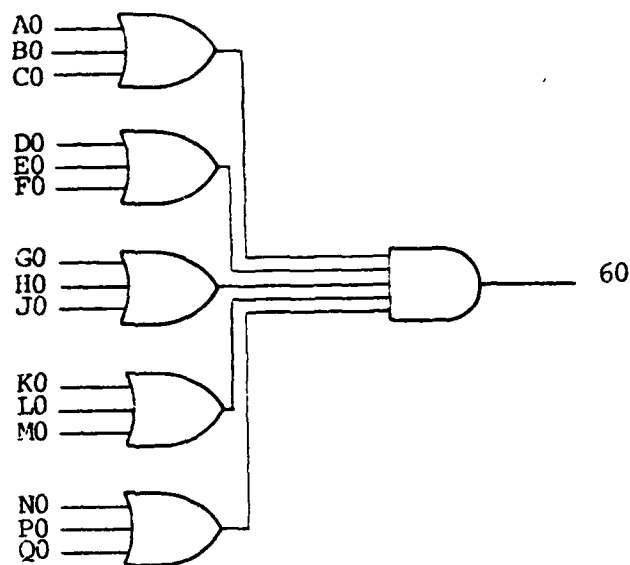
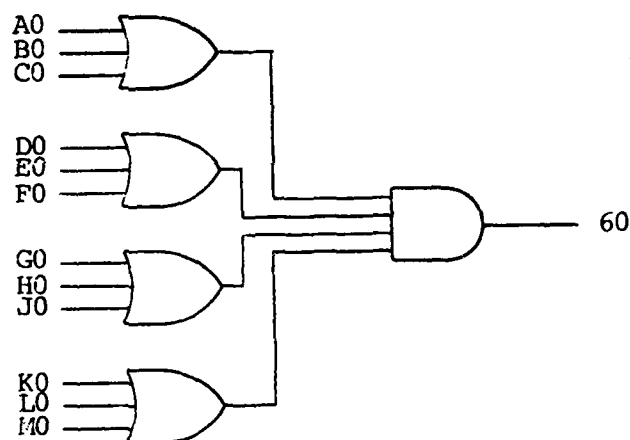


Figure 3.2.2-1 Logical Representation

CEC0
5-15 INPUT OR - AND GATES



CEC8
4-12 INPUT OR - AND GATES



CEC9
3-8 INPUT OR - AND GATES

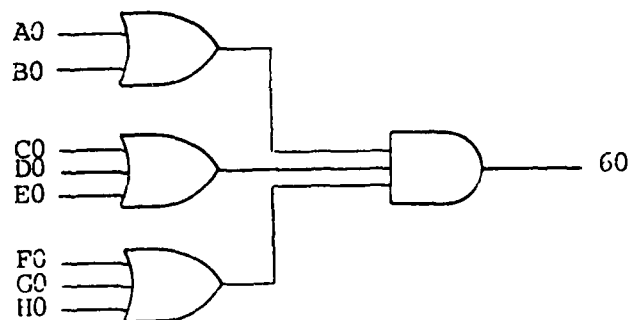


Figure 3.2.2-1 (continued)

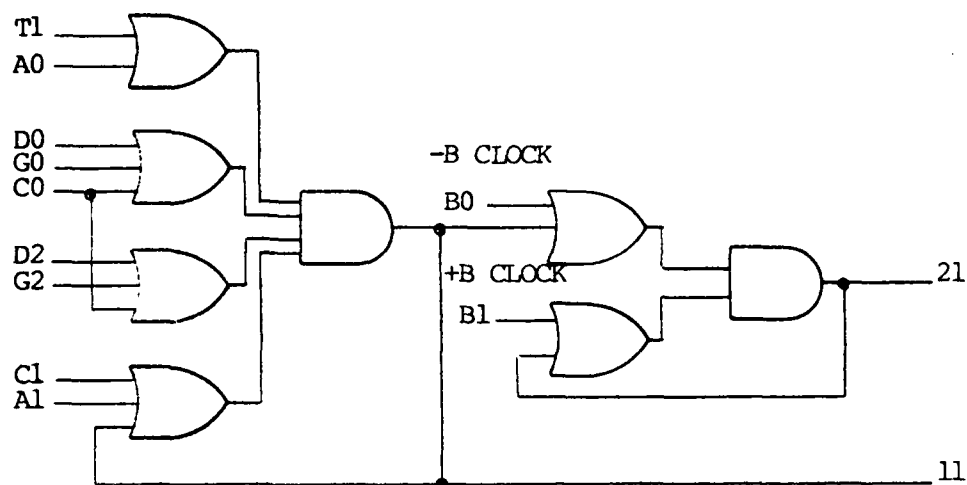
CED5
2 INPUT SRL

SCAN IN
-A CLOCK

DATA 0
-GATE 0
-C CLOCK

DATA 2
-GATE 2

+C CLOCK
+A CLOCK

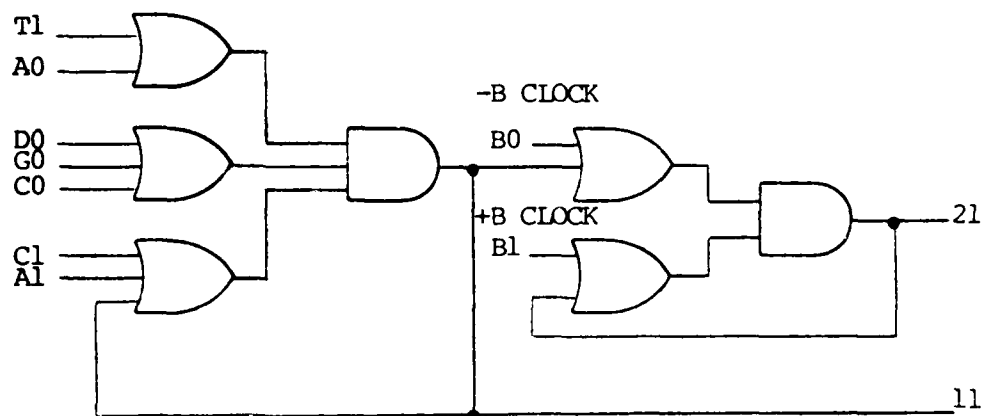


CED6
1 INPUT SRL

SCAN IN
-A CLOCK

DATA
-GATE
-C CLOCK

+ C CLOCK
+ A CLOCK



CTN0, CT19
CLOCK DRIVERS

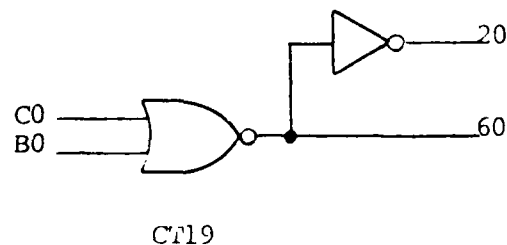
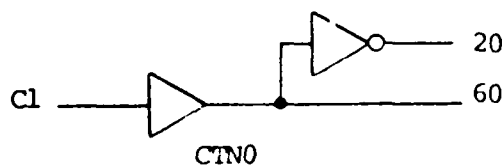
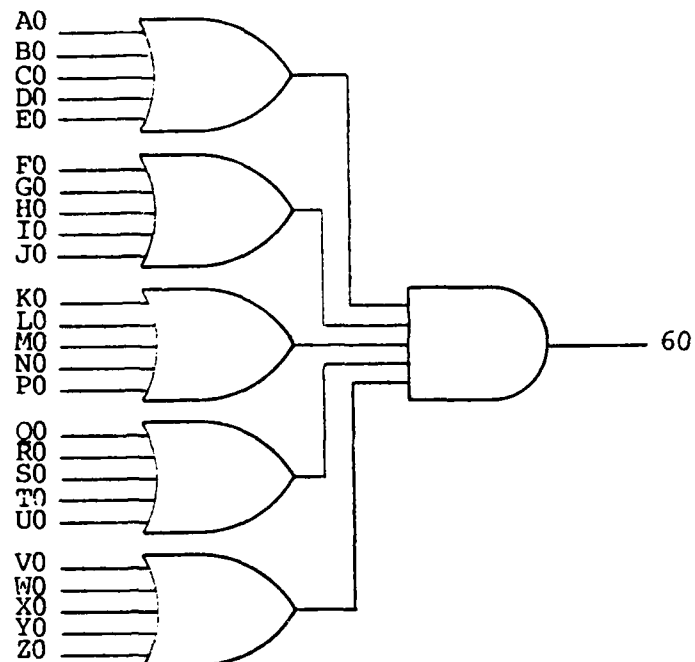


Figure 3.2.2-1 (continued)

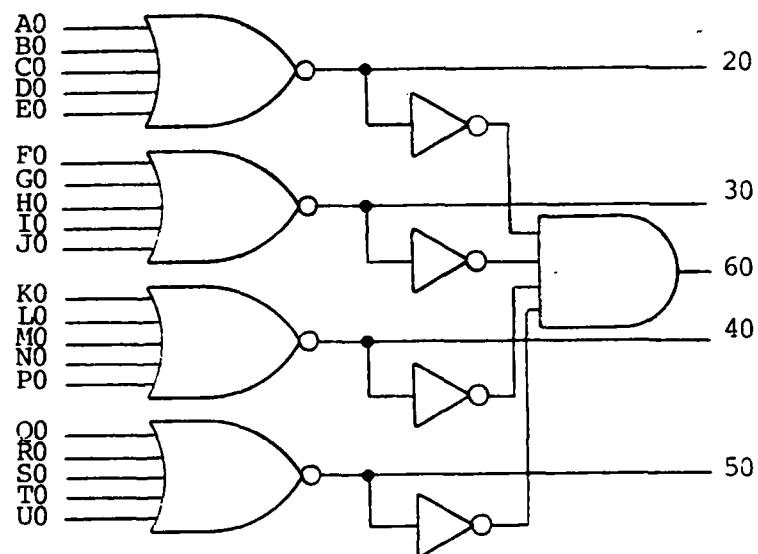
CED7

5-25 INPUT OR - AND GATES



CED8

4-20 INPUT OR - AND GATES



CEE0

2-10 INPUT OR - AND GATES

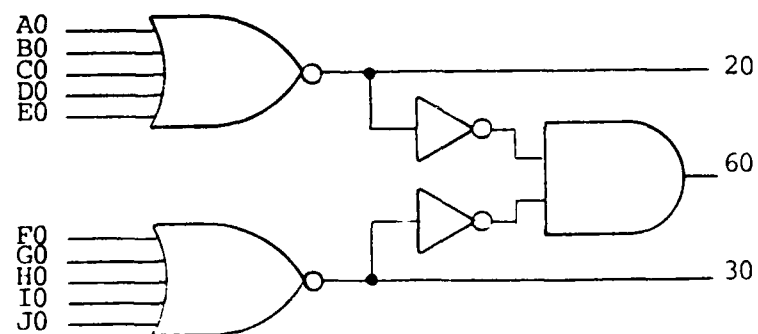


Figure 3.2.2-1 (continued)

A BDL/S statement defining a logic function, also defines the inputs to and outputs of that function. This means that the BDL/S statement can be represented pictorially as a block with input and output lines. This is, in fact, how the user's design is represented on an ALD (Automated Logic Diagram). The BDL/S statements can be generated and entered manually or the designer may have the ISSAC logic entry system generate the statements automatically.

3.2.4 INTERACTIVE SIMULATION AND ALD COMPOSING (ISAAC)

In order to increase logic designer productivity by enhancing logic entry, logic update, and logic checking, the ISAAC system (Interactive Simulation and ALD Composing) was developed within IBM/FSD. This system allows the designer to enter logic in an interactive mode from a remote terminal using the logic diagram which represents his design instead of writing individual BDL/S statements.

The designer simply enters the names of the logic books he wishes to interconnect and all the information necessary to describe that book in BDL/S is automatically retrieved from the book rule file. Once all the books are entered, the designer can then connect them together manually or with a light pen. Since the system is interactive, the designer sees the immediate consequences of his actions. Manual entry of BDL/S requires a batch job submission to obtain a logic diagram printout. Any errors must be corrected after these have been retrieved. In ISAAC, once the designer is satisfied with his design, ISAAC will automatically generate syntactically correct BDL/S statements.

3.2.5 AUTOMATED LOGIC DIAGRAM

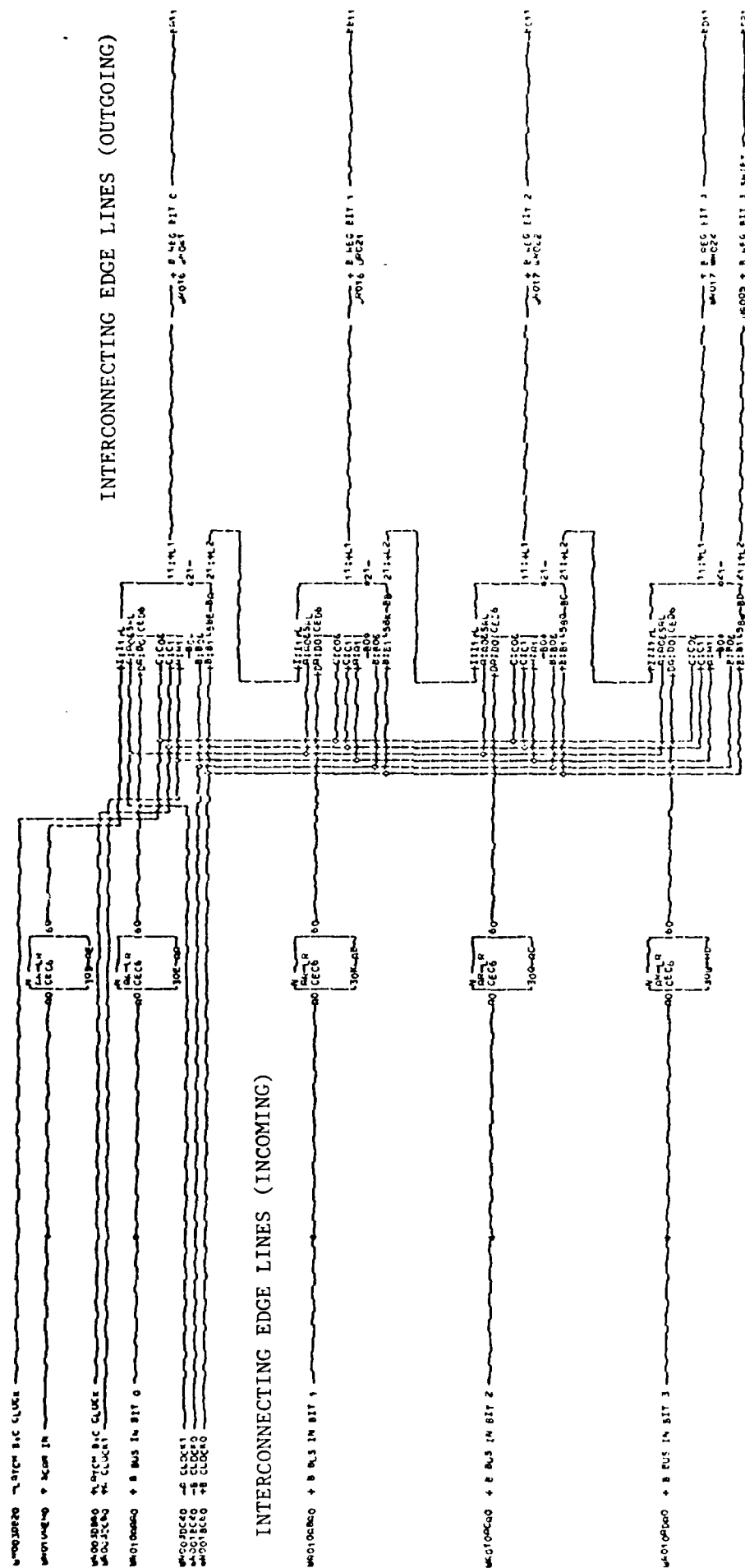
The ALD represents logic circuitry as a series of boxes connected by lines. Each box represents one logic or component function. These functions may be either high-level components (adders/ALUs, register, book, storage) or basic logic functions (ANDs, ORs, Exclusive-ORs). Figure 3.2-2 is an example of an ALD.

The ALD represents the user's input statements graphically, one-for-one, as blocks with defined input and output lines. Other information, such as the type of circuit or packaging and interconnecting data, appears either inside the block, or at the bottom of the ALD page. The ALD has established and easily-recognizable areas, which are described in the next few paragraphs:

The sheet title area provides information for the logic contained on that sheet. The ALD sheet name is unique to that particular sheet and is used as part of the line name on any other sheet whose lines connect the logic shown on this sheet. The title block itself contains identifying information that ranges from a short descriptive name of the functions shown in the logic area, to the physical location of that function in the actual hardware.

The diagram (logic) area, the principal part of an ALD, occupies most of the sheet. Each logic or component function is represented by a block. Inputs to the blocks are usually on the left side, outputs are usually on the right. In most cases, an ALD block represents a logical function. The blocks in the logic area may represent functions as simple as resistors or as complex as adders or storage devices. They may also contain comments pertaining to other logic on the same sheet. In any case, each block contains information defining its function, its location on the logic sheet, its physical location in the product being designed, and the inputs to and outputs from the function.

DIAGRAM (LOGIC) AREA



TITLE AREA

Figure 3.2-2 Automated Logic Diagram

SUPPORT AREA

The interconnecting areas, also called edge line areas, flank the logic area and define the interconnections between blocks that enter and leave the logic sheet. Since logic flow is generally from left to right, the left side is the input area and the right side is the output area. All cross-reference data between this ALD and others appears in the edge line area. Edge lines are usually the continuation of lines entering or leaving the diagram area. Interconnections are labeled according to their source or destination sheets, block, pin, and also a short title describing their function. Input lines entering the left-hand interconnection area may also be flagged (*1-7), meaning that the support area at the bottom of the sheet contains additional data describing that line.

The support area is located at the bottom of the sheet adjacent to the title block. Some of the information in the support area includes comments and physical description data. Comments provide general information about the data on the CID. The physical description data includes information about connections between physical components that are represented on the diagram.

The ALD in Figure 3.2-2 represents page WR010 from the SCS481 design. Two different book types are represented on the ALD, CEC6 and CED6. Each block contains information about the block. The top CEC6 is a block type N, meaning it is an elementary MS399 book. Its function is AR-LR or line receiver. The position on the ALD sheet in which it is printed is 30B and the block serial name assigned by the designer is AE. This CEC6 has one input A0 and one output 60.

Interconnecting lines are called nets and are named by concatenating the sheet name, the logic block serial, and the pin name of the source output. For example the output from the block just discussed is called WR010AE60. Each net can be assigned a signal name also. For example, the input to this block is called '+ Scan In'. The * on this net indicates more information on this net is in the support area. Examination of the support area indicates that this net is a primary input for the chip in physical location 1A-A1A1X01Y01AA01 and that it is the scan in pin for the first scan string (.KI001).

The title area informs the reader that this sheet represents the B input latches for the chip located at 1A-A1A1X01Y01AA01. This is a dummy location selected for the SCS481. The interconnecting edge lines (outgoing) region indicates underneath each signal name the pages on which these signals are connected to a sink.

ALDs are the primary design reference provided by EDS. The ALDs serve not only as a reference, but also as a tool for the system and logic designer. From these diagrams, the designer makes all logic updates and corrections.

3.2.6 MS399 STATISTICS PROGRAM

To aid the designer in doing some simple preliminary technology checks for MS399, a statistics program was developed. This program was designed to be generally applicable to all IBM bipolar masterslices. This program checks fan-in and fan-out restrictions, counts the number of internal cells used, estimates the number of primary inputs and outputs, and calculates power dissipation. Listings of book types used by ALD and by book type, book input/output pin counts, and net interconnection cross references are also available.

Figure 3.2-3 shows the statistical summary output from this program. Previous listings flagged any output net which exceeded the fan-out limit of the block driving it, as well as flagging any high powered blocks which were used where fanout does not require them. The statistical summary lists the total number of each book type used and the total number of internal cells used. The program even estimates the additional cells necessary to correct any books with excessive fanout. The summary also lists the number of connections on chip. The primary input and output estimation section is not operational here in Manassas which explains the negative numbers. The final figure listed is the estimated power dissipation for the chip designed.

STATISTICAL SUMMARY OF THE BOLS

TOTAL NUMBER OF BLOCKS BY BOOK TYPE

	CEB0	CEB2	CEB4	CEB5	CEB6	CEB9	CEC0	CEC1	CEC3	CEC5	CEC6	CEC7	CEC8
63	120	43	8	0	97	6	78	22	10	53	25	10	
LLC9	0	16	0	31	6	14	0	19	3	0	0	0	
	CEB3.8	CEB5	CEB5.A	CEB6	CEB7	CEB9	CEC0	CEC1	CEC4	CEC5	CEC6	CEC7	CEC8
LLC9	0	16	0	31	6	14	0	19	3	0	0	0	
	CEB3.8	CEB5	CEB5.A	CEB6	CEB7	CEB9	CEC0	CEC1	CEC4	CEC5	CEC6	CEC7	CEC8
63	120	43	8	0	97	6	78	22	10	53	25	10	
LLC9	0	16	0	31	6	14	0	19	3	0	0	0	

CTF9 CTN0 XE00 ELSE.

8 13 0 0

THE TOTAL NUMBER OF BOOKS IS 673

THE TOTAL NUMBER OF INTERNAL CELLS USED IS.....1307
 THE EXCESS FANOUT CELL PENALTY COUNT IS.....5
 THE EXCESS FANIN CELL PENALTY COUNT IS.....0
 THE INVERSION CELL PENALTY COUNT IS.....5
 CELL TOTAL.....1317

THE TOTAL NUMBER OF INTRA-BOOK CONNECTIONS IS.....0
 THE TOTAL NUMBER OF INTER-BOOK CONNECTIONS IS.....1749
 THE EXCESS FANOUT CONNECTION PENALTY COUNT IS.....10
 THE EXCESS FANIN CONNECTION PENALTY COUNT IS.....0
 THE INVERSION CONNECTION PENALTY COUNT IS.....5
 CONNECTION TOTAL.....1764

THE NUMBER OF PINS WITH EXCESSIVE FANOUT IS.....5
 THE NUMBER OF PIN TYPES WITH EXCESS FANIN IS.....0
 THE NUMBER OF TIMES BOTH OUTPUT PHASES EXIST IS.....0
 THE TOTAL # OF ELECTRICAL FANOUT LOAD UNITS IS.....2725
 THE TOTAL NUMBER OF INPUT PINS IS.....2899
 THE TOTAL NUMBER OF FANOUT PINS IS.....2582
 PRIMARY INPUT PIN COUNT.....429
 THE TOTAL NUMBER OF OUTPUT PINS IS.....849
 THE TOTAL NUMBER OF OUTPUT NETS WITH LOADS IS.....412
 PRIMARY OUTPUT PIN COUNT.....

Not Operational

STATISTICAL SUMMARY OF THE BOLS

AVERAGE AMOUNT OF POWER DISSIPATED IN MW.....1304

Figure 3.2-3 MS399 Statistic Program Output

The statistics program allows the designer to merely enter the logic and then power up or parallel those gates which exceed fanout restrictions. The internal cell count can warn the designer if he is coming too close to the upper limit of cells available. Likewise the power estimate assures him that he is not exceeding the upper chip power limit. The program eliminates a lot of error prone manual checking.

3.2.7 BDL/S SYNTAX CHECKER

Once the design has been entered either manually or through ISAAC and the designer is satisfied with the statistics program output, the BDL/S should be checked for syntactical correctness. EDS provides a foreground BDL/S Syntax Checker which creates a listing of any syntax errors on the display screen. Once all of these syntax errors have been corrected, the BDL/S is ready to be entered onto the CID or for other EDS processing steps.

3.2.8 SPECIAL LOGIC ENTRY REQUIREMENTS FOR I/O FLAGGING

Besides entering logic blocks and their interconnections, nets which cross to other ALD pages should be given signal names. Any unused input or output pins on blocks must be deleted so that physical design knows to eliminate the devices associated with that pin. All primary inputs and outputs must be flagged with a connector pin which has the same location number as the chip. All LSSD clocks, scan pins, and scan gates must be flagged to represent their function and polarity. All off chip driver nets are also flagged as such. Finally, any common I/O lines must be flagged so that the logic expander knows to create a dot-AND block with an output pin and a psuedo-input pin.

3.3 LOGIC ENTRY TO DATA BASE (NON-LSSD)

Logic entry for the non-LSSD versions of the SCS481 basically required substituting polarity hold latches and DFF'S for the SRL's in the design. This involved creating a logical representation of two DFF books, since none existed for MS399, and replacing the SRL book name with the DFF book name. The A and B clocks, as well as the scan input pins had to be deleted from the BDL/S description since the DFF's have no scan function. Also, the LSSD clock generation circuit was eliminated since the DFF only requires a single clock. This greatly simplified the clock circuitry, reducing it to only a powering tree.

The A and B latch SRL's had to be replaced with polarity hold latches. An additional logic representation for these polarity hold latches was unnecessary since the L2 latch portion of the SRL was functionally equivalent to the single polarity hold latch needed. The SRL book in the A and B latches were simply substituted with the L2 Book, and the extraneous clocks and scan paths were deleted.

An important thing to note about the non-LSSD design is that it is not directly analagous to the 74S481 design. First of all, the DFF model used is not the same as that in the 74S481, thus improving testability (Section 2.5). Secondly, the non-LSSD SCS481's contain non-LSSD versions of the enhancements. These enhancements probably did not directly affect testability except for the fact that they introduced additional logic to be tested. The stack design itself was probably less testable than any other section of logic. Thirdly, the non-LSSD SCS481 still had the B Bus Out port and OP9 out separate from their respective inputs. Also the OP8 disable improvement mentioned in Section 3.1.3.3 was included in the non-LSSD design. The breaking of these "feedback" paths may or may not have improved testability over the 74S481 design.

3.3.1 DFF BOOKS FOR NON-LSSD

Figures 3.3-1 and 3.3-2 show the logical representation of the two DFF books which were created for the non-LSSD design of the SCS481. Since these books had to be directly substituted for their SRL equivalents, common pin names and polarities were followed whenever possible. This explains why the L2 output was modeled as an inverted \bar{Q} of the DFF. Since the L1 output is unused in all applications for which these DFF's were to be substituted, a better model would have L2 be the Q output and L1 be the \bar{Q} output. Because this was not done, the testability of these latches were enhanced (see Section 2.5). Since these DFF's operate on the falling edge of the clock, inverters were used as clock drivers.

3.3.2 NON-LSSD DESIGN WITH SETS AND RESETS

In the hope of improving testability of the non-LSSD SCS481 design, set and reset lines were added to the DFF's. Figures 3.3-3 and 3.3-4 show the logical representations of the DFF books with sets and resets added. To incorporate these in the design a set and a reset primary input were added along with a powering tree for each to support the necessary fanout. Besides these changes, no other differences exist between this design and the non-LSSD design.

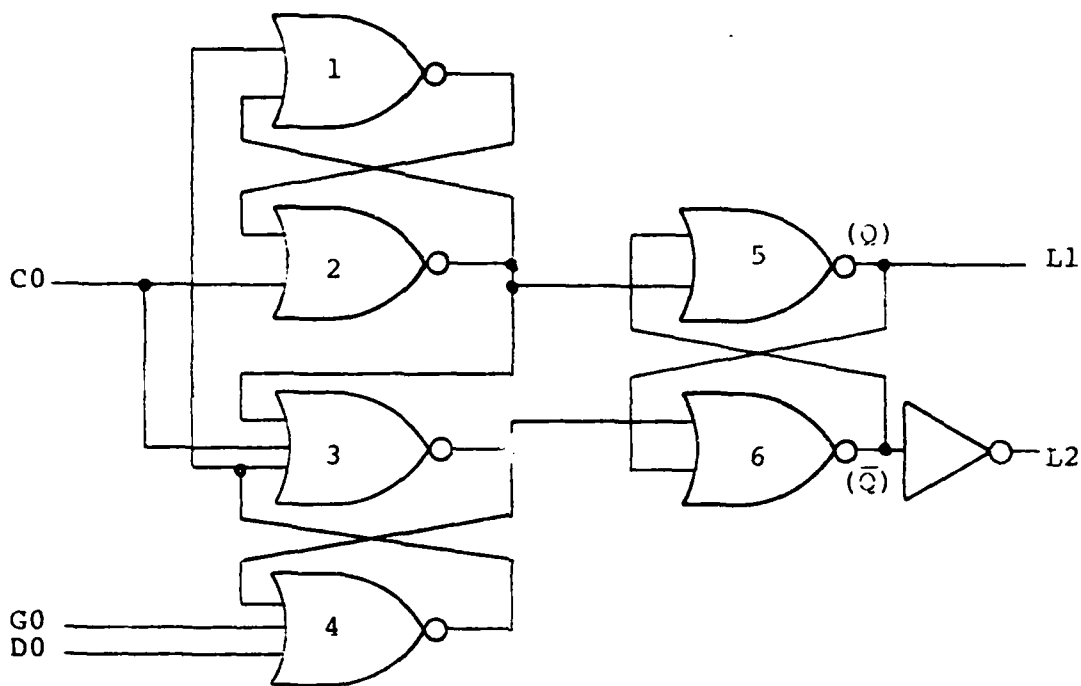


Figure 3.3-1 DFF To Replace CED6 SRL (Single Port) - D6SR

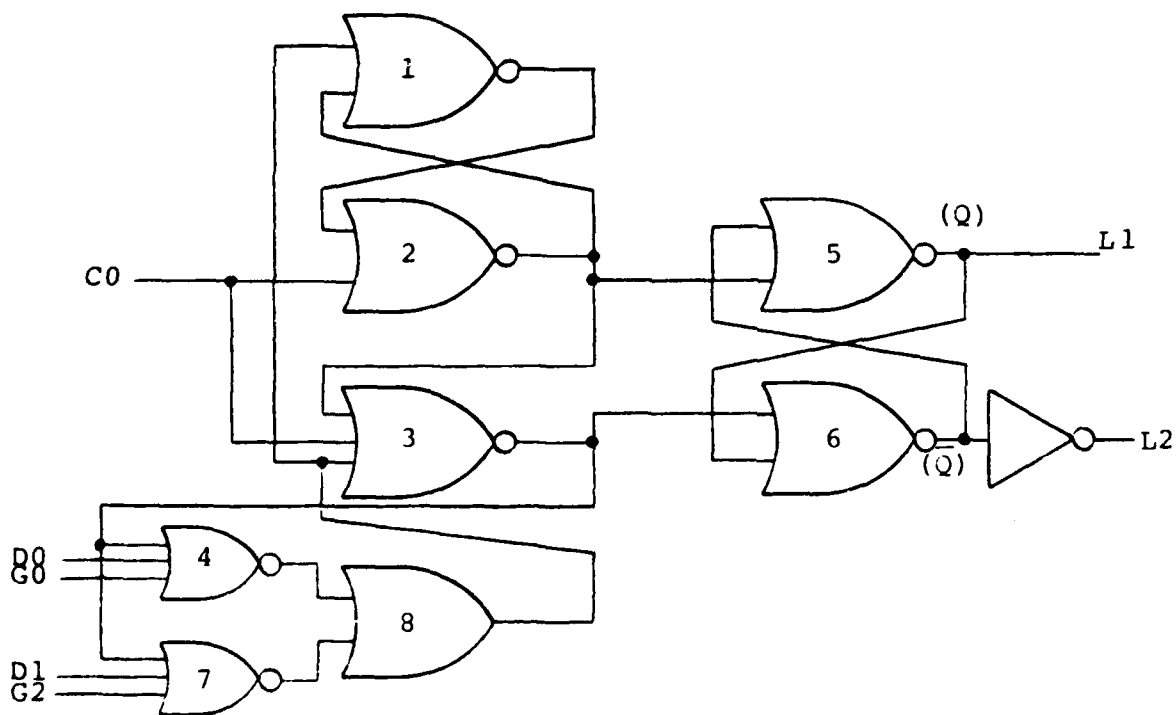


Figure 3.3-2 DFF To Replace CED5 SRL (Dual Port) D5SR

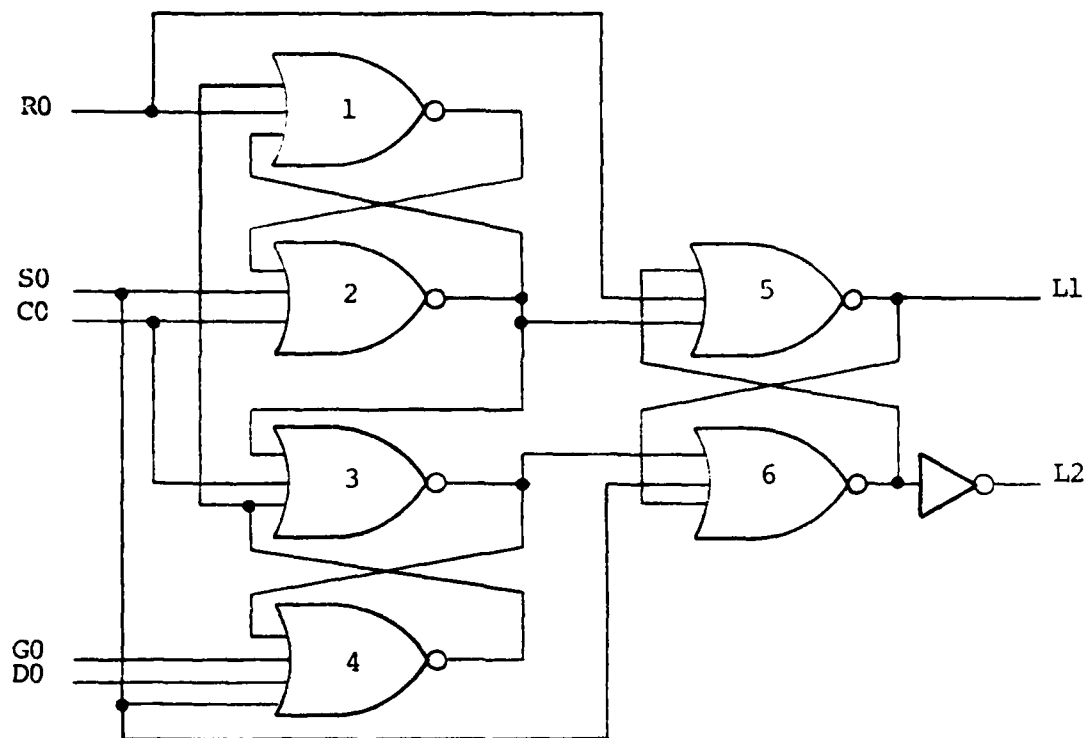


Figure 3.3-3 D6SR DFF With Set and Reset Lines

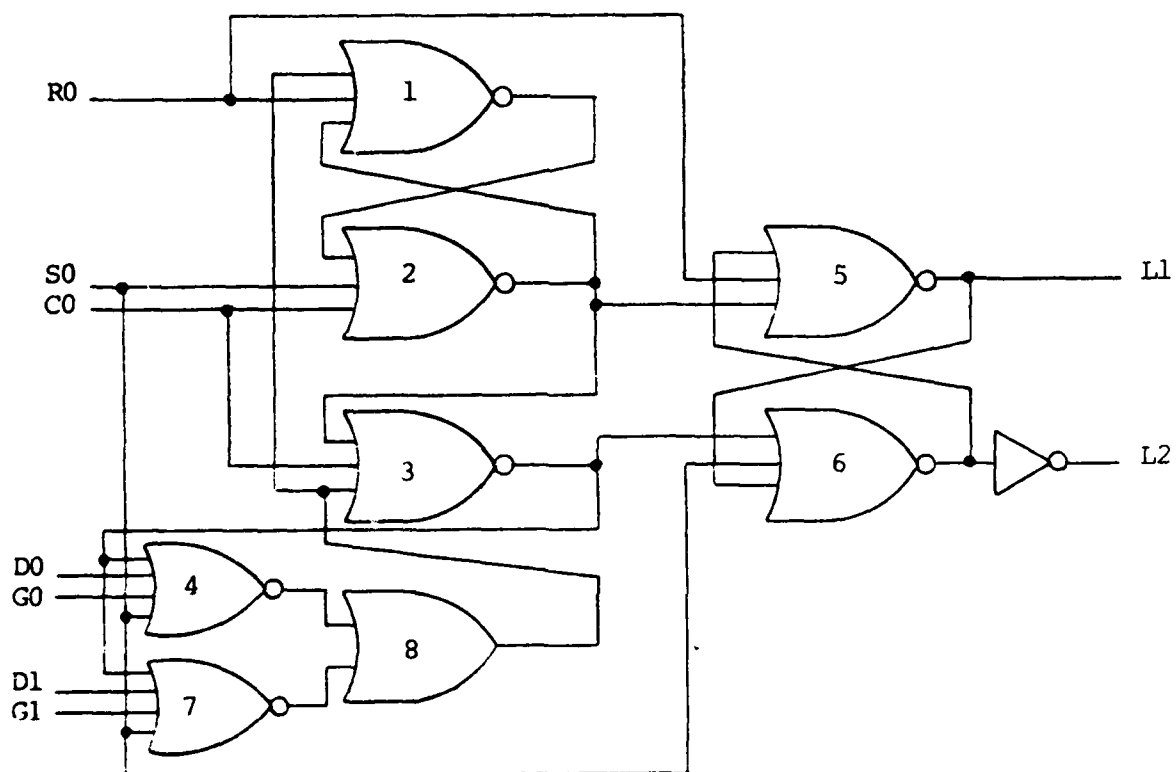


Figure 3.3-4 D5SR DFF With Set and Reset Lines

3.4 CREATION AND ENTRY OF FUNCTIONAL PATTERNS

Once the logic design has been entered into the system, the design can now be verified by the designer using the EDS Variable Mesh Simulator (VMS). Functional patterns must be written, and expected results calculated. Attempting to verify an entire LSI/VLSI chip simultaneously can become a mind boggling exercise; so, at least in the beginning, the design simulation concentrates on certain functional pieces. The designer works his way up to include all the logic on chip, if appropriate. How these patterns are written in EDS and the functional partitioning for simulation which was used are explained in the following sections.

3.4.1 BASIC DESIGN LANGUAGE FOR CONTROL (BDL/C)

Control is needed to tell the system how to simulate, how to design rules check, or how to run test generation on a design. The language is BDL/C, Basic Design Language for Control. BDL/C allows application of stimuli at specified times to the circuit, observe the responses calculated by the simulator, compare these responses to expected responses, and provide readable output such as timing charts.

The BDL/C defines the basic facilities where value changes are applied and observed. Facilities can be thought of as registers, buses, or even a single signal line. Any net in a simulation run whose logic state is to be changed, tested, or compared must be related to the bit position of a facility. Facility definition allows a large group of unwieldy net names to be assigned values by using an easily understood alphameric name. For example, the facility named OPBUS was 10 bits wide and stood for the ten op code primary inputs. The first step in writing BDL/C is to define all facilities.

Another important concept of simulation is simulation time. Simulation time is a relative time framework within which simulation events are scheduled to occur. A simulation event is a change in the logic value of a net due to either an explicit BDL/C statement or the

calculation of a response by the simulator. Simulation time is not established by a clock; it is determined by time fields specified in BDL/C statements and/or the simulator's scheduling of responses. Simulation time for a simulation run starts at zero and extends to some maximum specified time. Simulation time units are an indication only of the relative timing of events, and have no relation to actual time. The main points to remember about simulation time are that the executable BDL/C statements have time fields associated with them and the time fields determine the relative times at which events are to occur.

BDL/C also controls the type of simulation being performed; i.e., unit, nominal, maximum, or minimum delay. Additional BDL/C statements determine the format of the output timing charts. BDL/C can also perform comparisons on facilities and even print out an error message when a miscompare is discovered. In general, BDL/C is an extremely powerful language for simulation.

3.4.2 LOGIC PARTITION FOR SIMULATION

For the purposes of simulation six distinct logic partitions were identified on the chip. Identification of these partitions enabled the designer to focus attention on specific logic areas and simplify debug. However, the simulation patterns exercised the entire logic design and, in the final analysis, all data paths affected by a simulation pattern from input pins to output pins were verified as responding correctly. Table 3.4-1 shows a breakdown and brief description of the BDL/C patterns sets simulated on the SCS481.

3.4.2.1 Arithmetic Logic Operations

This partition consisted of the ALU and comparator, Sum Bus shift mux, the A and B input ports and their respective latches and multiplexers. Since this partition corresponds to the test site for which a RIT was generated in Rochester, much of the design verification for this logic section was done using the test site design data.

Simulation for this partition verified correct operation of all of the basic arithmetic and logic operations as well as the shift operations and status indicators. The input arguments were obtained from the A and B input ports and the results were observed at the Sum Bus.

The specific functions to be tested were those used in the first nine Operation Forms (the microprogrammable Op. Forms). These include: Add, Subtract, AND, OR, Exclusive OR, NAND, NOR, Exclusive NOR, NOOP, Compare, Left and Right Shift Logical and Left and Right Shift Arithmetic.

3.4.2.2 Control Logic

After the ALU had been successfully simulated and debugged, the control logic was exercised. In this partition primary emphasis was on the random logic which replaced the Micro Decode Logic Array. The simulation concentrated on the first nine Op. Forms in order to confirm that the correct arithmetic/logic operation was selected. Those fields in the Op. Forms which select source and destination were exercised sufficiently to verify that the control lines select the correct arguments or combination of arguments as in the case of the B Port where the B input can be ANDed with one of three registers before being operated on by the ALU. Because some of the status indicators are functionally dependent on whether the chip is in the least significant, intermediate, or most significant position, it was necessary to write patterns to specifically exercise these position dependent pins in such a way as to get uniquely different values depending on which position the chip is set for via the position pins.

3.4.2.3 Address Registers

The Program Counter and Memory Counter can be effectively isolated from the ALU and control logic and were simulated after the ALU and control logic have been debugged. Simulation patterns were generated to load and increment the PC and MC using the asynchronous controls. The \overline{CCI} and \overline{CCO} were observed with particular attention given to the MSP to verify that the

Table 3.4-1. BDL/C Dataset Description For Simulation

BDL/C Name	Description
ADDALL	Tests OP Form IA,B (ADD) all possible A,B,CIN combinations
ADDSHIFT,1-6	Tests OP Forms IIA,B,III (ADD & SHIFT) with check of shift bits in and out
ALU 1&2	Tests load WR, OP Form IA,B (ADD) all A&B MUX combinations
AUTO,2	Same as ADDALL with automatic check of X,Y,EQ,OVFL,AND COUT, etc. AUTO2 automatically checks AG,LG,EQ,OVFL,& COUT in MSP
COMPARE	OP Form VIIA,B (COMPARE OPERATION) Check
COUNTMC,PC	PC,MC increment function check
CRC,1	OP Form X (CRC) check for all package positions
DATAOUT	Address and data out port checks
ENHANCE 1,2,3	Enhancements test
LOGICAL	Checks OP Form VIII A,B,C (LOGICAL OPERATIONS)
NOOP	Checks All OP Form IX (NOOP) combinations
SBOUTS	Checks OP Forms IV-VI (SHIFTS) for proper shift bits out
SHIFT 1-5	Extensive check of OP Forms IV-VI (SHIFTS) for all package positions and shift forms
SIGNDIV,1-3,T	Tests Op Form XI A,B,C,D,E (SIGNED DIVIDE) for all flowchart paths and package positions
SMPY 1,2	Tests OP Form XIV (SIGNED MULTIPLY) for all flowchart paths and package positions
TEST	Check out test circuits
TSTOUT,1-5	Extensive check of X,Y,AG,LG,EQ,OVFL,& COUT for OP Forms I-III
UNDIV 1,2,T	Tests OP Form XII A,B,C (UNSIGNED DIVIDE) for all flowchart paths and package positions
UNMPY 1,2	Tests OP Form XIII (UNSIGNED MULTIPLY) for all flowchart paths and package positions

CCO/Overflow pin was used by the ALU for overflow indication. Operation of the Address Out Port and Mux as well as the AO Mux Select asynchronous control line were confirmed at this time.

3.4.2.4 Data Registers

The fourth partition consisted of the Working Register, Extended Working Register and miscellaneous data paths which tie the partitions together. Single and double length shifts involving the WR and XWR were performed. Arithmetic, logical and circular shifts were performed both to the left and to the right. Those Op Forms which perform Arithmetic operations followed by shifts were executed. The position controls were exercised in order to confirm that the high order bit behaves correctly when the chip is set to MSP.

The simulation at this time also exercised the asynchronous controls in order to demonstrate proper flow of data within the chip and correct selection of source operands for the Data Out Port (DOP).

3.4.2.5 Multiply, Divide, CRC

After all of the logic in the preceding partitions had been verified, the controls for multiply, divide, and CRC (cyclic redundancy check) were simulated. Each of the five Op Forms X through XIV were verified with the chip position controls set for LSP, IP, and MSP. Particular attention was given to Op code pins Op 8 and Op 9 to insure correct behavior since these pins may be input or output depending on the position setting. By using BDL/C the various shift bits could be tied together on the single chip to allow simulation of four bit multiply's and eight bit divide's. This proved very helpful in debugging the macro command. It also demonstrated that the SCS481 obtained the proper results even though the shift bits were inverted.

3.4.2.6 Enhancements

After simulating the standard 74S481 functions, the enhancements were simulated to verify their design. The stack and breakpoint register were exercised in conjunction with the program counter and memory counter to confirm proper operation. The hold A and B latches and memory counter thru A mux controls were also simulated. Finally the test circuits were verified to guarantee the proper function of the ring oscillator and other test circuits.

3.4.3 SUGGESTED IMPROVEMENTS TO DESIGN VERIFICATION

Several procedural changes would have improved the effectiveness of design verification. First, and most importantly, the person(s) writing the functional patterns should be completely insulated from the logic design. Both the logic designers and pattern generators should work from the same specifications, but not have significant interaction. Any deviation from expected results during simulation will then either result from an error in the logic design or an incorrectly calculated expected result. The odds that logic design errors similar to those in the SCS481 will occur are greatly reduced since both the logic designer(s) and pattern generator(s) will probably not make the same misinterpretation of the specification. When the logic designer(s) and pattern generator(s) are one and the same, no check and balance system exists for an error of this type.

Secondly, the design verification patterns should be written so that as little manual intervention is required to check expected results. The ultimate solution is a completely self checking pattern set similar to the Auto and Auto 2 pattern sets. In these sets, the BDL/C automatically calculates expected values for all the addition combinations, applies the necessary patterns, checks the actual results, and prints out an appropriate error message if necessary. Pattern sets such as these have the advantage of eliminating human error since all expected results are algorithmically generated and automatically checked. They can also check a greater number of conditions in far less time than can be done manually.

Finally, each simulation run should be followed by the EDS design verification measurement program (DVME). This program keeps a running total of all nets in the design which have not been conditioned to a '1' and a '0' as well as which gate inputs have not been conditioned in a sensitized condition. By examining this list of nets, the pattern generator can then determine what areas of the logic should still be exercised.

3.5 DESIGN VERIFICATION (LSSD)

Once functional patterns have been created and entered into EDS via BDL/C, these patterns must then be simulated. The designer uses the Variable Mesh Simulator (VMS) as well as several other EDS programs to accomplish the design verification. An explanation of those programs and their importance to design verification follows.

3.5.1 EXPAND ACTION

In order to use the logic entered via BDL/S (Section 3.2.3), the design verification and test generation (DV/TG) section of EDS requires that the logic be expanded. Expanding logic involves transforming macro-representations into one or more constituent micro-blocks and adding information about the blocks that is required for subsequent applications. There are two basic types of expansion: design and non-design.

Design expansion is associated with the physical design of a part. It transforms a macro-block representation into a more detailed representation. It allows convenient representation of logical entities as macro-blocks that can be design expanded either into physically implementable entities for Physical Design or representations on which subsequent non-design expansion can be performed for DV/TG. An example of such a logical macro is the SRL book which is made up of an L1 book and an L2 book.

Non-design expansion is associated with the logic design of a part. Like design expansion, non-design expansion can transform macro-blocks into micro-blocks. In addition, however, it can also transform logic macro-blocks to their constituent primitive functions that can be used by the DV/TG applications. It is an expansion to primitives at or below the packaging level. Information that is required for modeling the logic for DV/TG is added from the logic books. Because some DV/TG applications may require different models of the same logic, non-design expansion can produce different micro-representations of blocks if alternate expansion information is provided in the book. Each possible representation is identified as a unique expansion type.

Generally speaking, the two processes differ in that design expansion deals with blocks representing physically implementable devices (chips, cards, circuits, etc.) to create representations for Physical Design. Non-design expansion deals with blocks that represent physical units as well as blocks that represent primitive logic below the lowest physical level and creates representations for DV/TG.

The expand action also provides a preliminary check on several elements of the logic design. The expander checks to make sure that any unused pins on logic books have been deleted and that all pins which were deleted were pins flagged as removable by the logic book rules. This allows the designer a basic check that the book was at least connected (or not connected) properly. The expander also prints a list of those nets it finds flagged as primary inputs, primary outputs, and common I/O. The designer can then compare this list to his own to check that all I/O nets have been properly identified.

3.5.2 SIMULATION

Simulation is performed using VMS. VMS has several options for output and simulation mode which are selected by the BDL/C. An example of a normal timing chart output is shown in Figure 3.5-1. This particular output shows that the addition of $1+F+1$ eventually equals 2 after the signals propagate through thirty-two units of delay (from time 107 to time 139).

In logic design, delay refers to the time required for an electrical impulse to travel between two points on a circuit. Although a more detailed sub-division is possible, circuit delay is usually divided into two types: block delay and net delay. Block delay is the lapse between the time a block receives an input, and the time it produces the appropriate output. Net delay refers to the time required for a signal at the output of a block to travel through a conductor and reach an effective level at the input to another block.

When working with a hardware model, circuit delays are an inherent physical characteristic. The software simulation, however, has no such inherent characteristics. The designer must somehow indicate to the system what delay factors are to be considered. Depending on the degree of accuracy required, there are several ways to do this.

3.5.2.1 Unit Delay Simulation

The simplest and consequently the least accurate delay values are unit delays. Unit delays are provided by the individual DV/TG applications. When working with unit delays, the application simply assumes that each circuit element has the same delay of one "unit." Unit delays are applied only to blocks; net delays are commonly not considered. This mode of operation will provide gross function testing, but the lack of actual timing information may create results of limited usefulness. Unit delay simulation was the primary VMS mode used in DV of the SCS481 and is shown in the Figure 3.5-1.

3.5.2.2 Nominal Delay Simulation

Nominal delays are calculated "best guesses" about how individual circuit elements will behave in a real situation. Such delays are derived with knowledge of the circuit type and the interconnections involved. Default values for wiring capacitance are assumed. DV/TG operations performed with nominal delays are generally closer than unit delays, but still may not precisely predict the operation of the physically designed circuit and are generally pessimistic since a relatively high default capacitance is assumed.

3.5.2.3 Calculated Delay Simulation

Calculated delays are the most accurate possible prediction of how a unit will operate once it has been manufactured. They are provided by a system function called the delay calculator that takes into consideration such design elements as wiring, off-unit connectors, and fan-in and fan-out situations. Calculated delays may be derived at any time in the design effort, but are most effective when they are created from all the data available for a fully designed package. When the delays are calculated after the completion of physical design, they are called "as wired" delays. Consequently, they find significant use during delay path verification (Section 3.8).

3.5.3 ALL EVENTS TRACE (AET)

One of the outputs from VMS is an all events trace (AET). The AET contains all stimuli, and the response of every circuit net to those stimuli, stored in time sequence. This information can be used to print timing charts like Figure 3.5-1. The AET is also used by other EDS jobs downstream such as DVME (see 3.4.3).

3.5.4 TIMING ANALYSIS (TA)

Once either nominal or as wired delays have been calculated, the Timing Analysis (TA) program can be used. The TA application within EDS provides an automated design tool which performs fast and accurate analysis of the timing relationships in the logic.

The TA process takes into account circuit delays and physical delays for both turn-on and turn-off, as well as their statistical variations. The process considers the interactions between data and clock signals in the storage elements in order to determine early and late signals.

TA provides a reasonably quick and accurate analysis of critical time paths as well as indicating potential fast path problems. Since TA is very delay dependent, the most extensive analysis is conducted after physical

design during delay path verification (Section 3.8) when as wired delays can be used. Since the SCS481 was only a map-over of an existing design into a faster technology, a cursory examination of the as wired delay calculation for known critical paths was felt to be sufficient to assure proper timing; therefore, TA was not used.

3.5.5 LSSD RULES CHECK

Early in DV the design should be examined by the LSSD rules checking program. This allows rules violations to be detected and corrected before extensive DV has taken place which might be invalidated by any logic changes. Sometimes the rules checker will even uncover design errors (Section 3.1.3.3).

Since the rules checker uses VMS, the logic must have been successfully expanded prior to using the rules check program (Section 3.5.1). Using algorithms described in recent literature* and information gleaned from connector flags entered during logic entry (Section 3.2.8), the rules checking program generates BDL/C to confirm conformance with the LSSD design rules (Section 3.1.2). Once the logic is expanded and the BDL/C has been generated, the BDL/C is simulated. Every time unexpected results are obtained in the logic, an error is flagged listing the invalid results. Successful completion of the rules checker requires that no error messages be printed.

3.5.6 TESTABILITY ANALYSIS

Provided the design has passed the rules checking program just described, the design is virtually guaranteed to be testable; therefore, a testability analysis per se is not needed. However, since LSSD has greatly reduced

*"Automatic Checking of Logic Design Structures for Compliance with Testability Ground Rules" H. C. Godoy, G. B. Franklin, and P. S. Bottorff pp. 469-478. IBID

the cost of test generation, common practice is to run test generation as a preliminary testability analysis. This shows that an acceptable test percentage is obtainable and will also flag many of the redundant faults (Section 3.9.3.3). These redundancies can then be removed before physical design begins, improving the final test coverage. A preliminary test generation run was made on the SCS481 with a 98% test coverage.

3.5.7 LOGIC UPDATE

Logic update is required anytime a logic error is encountered using one of the previous tools. Logic update uses the same methods as logic entry to make corrections to the logic, manual BDL/S entry or ISAAC (Sections 3.2.3 and 3.2.4). Before DV can use the corrected logic, it must be re-expanded (Section 3.5.1). In essence DV is an iterative process with logic entry until the designer is satisfied that all errors have been detected.

3.6 DESIGN VERIFICATION (NON-LSSD)

DV for the non-LSSD versions involved only unit delay simulation with manual comparison of the results with the identical LSSD unit delay simulation. To accomplish this, all nine facilities defining the additional LSSD primary inputs were deleted from the BDL/C. A set and a reset facility were then defined and initialized at a '0'. Since the nine LSSD pins were not functionally exercised by any of the pattern sets except for being initialized in a non-controlling state, their deletion did not affect any of the pattern sets as written.

All of the BDL/C data sets in Table 3.4-1 except TEST were simulated against the expanded non-LSSD version of the logic and compared to the LSSD version. The only noticeable difference between the two designs was the disappearance of the latch glitch on the clock edge in the non-LSSD design. Figure 3.6-1 is the non-LSSD timing chart version of Figure 3.5-1.

[illegible][illegible][illegible]

Figure 3.6-1 Non-LSSD Unit Delay Simulation Output

3.7 PHYSICAL DESIGN

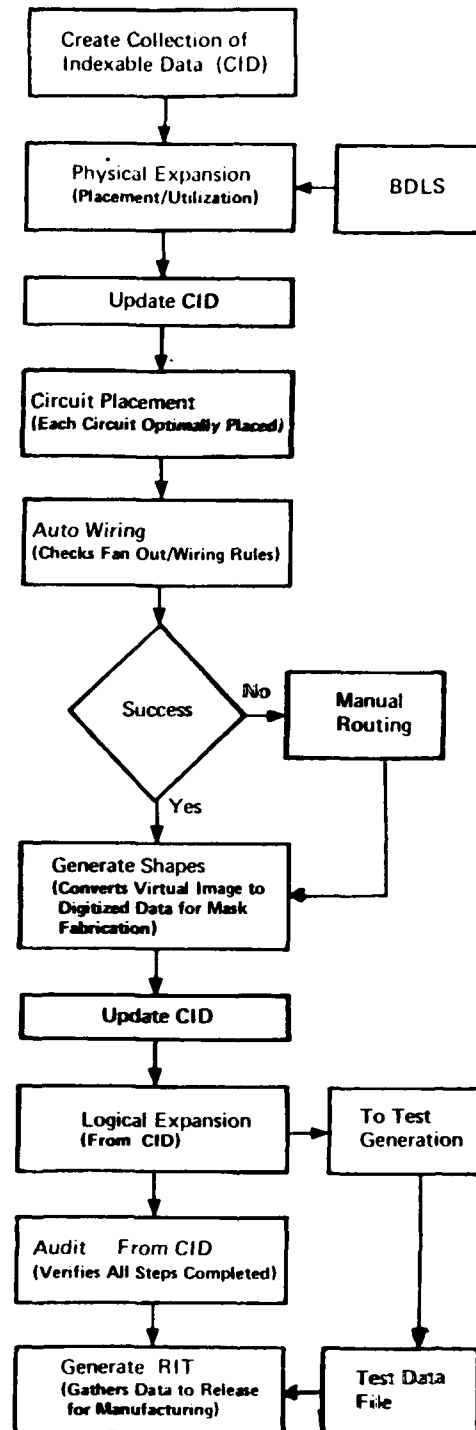
After the logic design has been initially recorded and simulated, the user can then translate his logical design to a physical design by adding additional nomenclature on the CID. During this cycle, the user can process application programs--called "actions"--in the EDS Physical Design subsystem. Again, the design can be simulated and tested using VMS. This is called Delay Path Verification and is described in the next section. The status of the design is checked to ensure that it is complete and that technology constraints have been followed; bill of material and part number data are also added.

The Physical Design subsystem includes a large variety of checking, packaging, design aid, wiring, and listing functions. The programs help the user arrange the logic on a physical package; wire up the unit, and check the design. The development of the user's design includes: pin assignment, net ordering, wiring, design checking, and CID updating with completed design data.

From the beginning of the actual design process, the designer is affected by the limits of the technology. It is at this point in the process, however, where all incompatibility between logic design and technology must be resolved. The logic design is converted to physical design descriptions that are intelligible to a manufacturing process.

Each logic function becomes part of a physical device. These devices must then be placed on a physical unit, wires must be routed, physical input and output pins must be provided, shapes must be generated for the metallization and all of these processes must be checked for correctness.

Physical design is performed after the logic design of the machine has been generally completed and stored in the design file, or CID. The CID then contains the logic design of the hardware as well as certain physical characteristics that the user may optionally supply. After physical design is completed, the majority of the physical and logical information resides on the CID. The typical physical design and release methodology is shown in Figure 3.7-1.



PHYSICAL DESIGN/RELEASE

Figure 3.7-1

3.7.1 PHYSICAL DESIGN TECHNOLOGY RULES

The CID contains only a representation of the designed machine. It does not contain the guidelines or rules for generating machine configurations. Since these guidelines are constant for a machine technology, a new dimension must be added to the design process: technology rules.

A typical rule is a table which describes a particular aspect of the physical package. An example of a rule would be a table containing the names of all possible types of wire that may be used on the machine. PD rules may be divided into three basic categories: geometry rules, relationship rules, and constraint rules.

Geometry rules describe the shape, size and internals of the parts of the machine. For example, if a rectangular part has holes running through it in a regular grid pattern, this characteristic must be described in a geometry rule.

Relationship rules describe how several parts of the machine may be assembled together. If four small parts can be plugged onto one larger part, this characteristic must be recorded in a relationship rule.

Constraint rules dictate criteria to be used by PD programs in completing physical design work. If the longest wire that may be placed on a machine part is two inches, this fact must be specified as a constraint rule. PD uses the rules in conjunction with other physical design data in order to achieve desired physical design work.

3.7.2 PHYSICAL DESIGN ACTIONS

The Physical Design portion of LVS performs three basic functions for the designer: placement and wiring of components; checking of the work done; and listing data.

Placement is the positioning of components. An example of placement are the positioning of logic circuits on a chip. The placement function determines the best possible arrangement of the components.

Wiring is the interconnecting of the components with printed or embedded wires. For example, once the logic circuits have been placed on the chip, wiring automatically determines how to connect them so they reflect the logic design.

Placement and wiring comply to the rules of the technology which is being used.

PD also performs checks of the physical design to assure no error has occurred. This is particularly useful on parts of the physical design which may have been done manually. An example of a PD checking function is Location Check. It checks to ensure that one physical location has not been used more than once.

The final function of PD is the Listing Function. Net List is an example of a PD Listing function. It lists the information associated with each pin belonging to a physical net.

The PD functions are accomplished by sets of actions. Each action is identified by a mnemonic. To perform a task, such as wiring an MS399 chip, a series of PD actions would be run. The action AWIRE (Automatic Wire Routing) would certainly be one of the actions in the series.

3.7.3 MS399 PHYSICAL DESIGN

A description of the PD actions for the Physical Design of the SCS481 and their scheduled and actual completion dates are listed in Table 3.7-2. The delay at steps four and five, PDDEL and ASSIGN, was caused by back-dated technology rules for some of the newer MS399 book types. The latest release for these rules had to be obtained from other IBM locations before Physical Design could continue.

Table 3.7-2. MS399 Physical Design Steps

Job Step	Sched Comp	Status	Comp	Result	Function
1 EFU1	12/27/79	TRIED	12/26	SUCCESS	CREATE CID
2 LPULA	12/27/79	TRIED	12/26	SUCCESS	LOAD BDLS TO CID
3 LPUU	12/27/79	TRIED	12/26	SUCCESS	LOCATION CODE
4 PDDEL	1/2/80	TRIED	1/9	SUCCESS	PHYSICAL EXPAND
5 ASSIGN (PD)	1/2/80	TRIED	1/15	SUCCESS	PIN ASSIGNMENT
6 PLACE1 (PDCW)	1/7/80	TRIED	1/16	SUCCESS	AUTO PLACEMENT
7 PLACE2 (PDCW)	1/10/80	TRIED	1/16	SUCCESS	AUTO PLACEMENT
8 AWIRE (PDCW)	1/16/80	TRIED	1/17	SUCCESS	AUTO WIRE
9 MROUTE (PD)	1/23/80	TRIED	1/21	SUCCESS	MANUAL WIRE
A SHAPES (PDSHP)	1/30/80	TRIED	1/23	SUCCESS	SHAPES GEN/CHECK
B FDBKI (PD/LPD)	1/31/80	TRIED	1/23	SUCCESS	FEEDBACK TO CID
C TESTGEN	2/7/80	TRIED	2/1	SUCCESS	ENTIRE TESTGEN
D AUDIT (PD)	2/13/80	TRIED	1/24	SUCCESS	AUDIT CHECK
E FDBK2 (PD/LPD)	2/14/80	TRIED	1/24	SUCCESS	FEEDBACK TO CID
F RIT	2/18/80	TRIED	2/4	SUCCESS	RIT GENERATION

Step C represents the entire test generation procedure described in Section 3.9. As explained, all five of the test generation steps from expansion to automatic test generation were submitted sequentially overnight. Once the test coverage was confirmed, the last two steps, which generated the Test Data File (TDF), were executed sequentially. No knowledge of the design or test generation was required.

Steps D, E, and F are all necessary for the Release of Data to Manufacturing explained in Section 3.11. Steps 1 through 3 are described in the following subsections.

3.7.3.1 CID Initialization

The first three job steps involve the initialization of the CID. In these steps the CID is created, the design is entered onto the CID, and all necessary design identification is placed in the CID control data set. Once the CID has been created, the BDL/S that has been entered by the designer can then be loaded onto the CID. Following that, the location, part number, part name, technology type, and other pertinent information about the design is entered into the CID control data set. With the CID initialization complete, design verification, physical design, and test generation can all use the information to perform their various tasks.

3.7.3.2 Physical Design Expansion

Step four, PDDEL, is the design expansion which was described in Section 3.5.1. This expansion is different from the expansion used in design verification in that nothing is expanded beyond physically implementable circuit books. For MS399, this means that the SRL's are expanded into L1's and L2's, but not to their elementary gate functions. Expansion to the smallest physically implementable blocks is necessary for efficient functioning of the placement and wiring programs.

3.7.3.3 Placement

The Physical Design automatic placement of the logic circuit books is accomplished by steps 5, 6, and 7. The ASSIGN action begins by allocating off chip drivers and on chip receivers to I/O pins and circuit nets. Using this information, the remaining logic circuit books are placed into the masterslice cells using actions PLACE1 and PLACE2. Placement is made in an attempt to minimize global wire lengths wherever possible. Figure 3.7-3 shows a printout of the placement for the upper left hand corner of the SCS481.

3.7.3.4 Wiring

Once the logic circuit books are placed, the automatic wiring action, AWIRE, is run to connect the books together. Any nets left unconnected by AWIRE must be manually routed using the MROUTE action. Only sixteen nets were incompletely wired by AWIRE for the SCS481.

The wiring programs also check block fan-out and total net capacitance. Any nets with greater than 3pf are flagged and must be checked by the logic designer to assure that the additional delay incurred will not adversely affect the design. Four nets exceeded the 3pf limit in the SCS481. Their impact is discussed in Section 3.8.2. An output representing one level of the wiring for the upper left hand corner of the SCS481 is shown in Figure 3.7-4.

3.7.3.5 Shapes Generation and Check

The previous wiring and placement actions used only logical representations of the MS399 chip. The SHAPES action takes these logical representations along with some technology rules and generates descriptions of the actual shapes of the metal interconnect lines which will be used to make the masks. These shapes are described using IBM's graphic language. Once these shapes have been generated, they are checked by SHAPES to assure that all

122

2040529
10-01-1001Y01-0
HH2930

123

Figure 3.1-4 Wiring Printout

the technology rules have been followed. This step discovered a wire which was routed over a blockage inside a logic book. Manual re-routing was able to solve the problem.

After successful shapes generation and check, the Physical Design process is "fed back" to the CID using action FDBK1. Physical Design data is now ready for release and test generation can begin. Table 3.7-5 gives a summary of the results after Physical Design, including circuit and I/O usage, placement statistics, and wiring results.

Table 3.7-5. MS399 Design Statistics

Circuit Statistics

USED/AVAILABLE

ON CHIP RECEIVERS	56/88
INTERNAL CELLS	1307/1496
OFF CHIP DRIVERS	35/60
I/O	86/94

Placement Statistics

TOTAL NETS	960
TOTAL CIRCUITS	741
TOTAL I/O	86
PREPLACED CIRCUITS	741
TOTAL CELLS	1398/1650

Wiring Statistics

NETS COMPLETELY WIRED	914
NETS INCOMPLETE OR UNWIRED	16
CONNECTIONS REMAINING	17
PARTIAL INPUT NETS WIRED	43
NETS EXCEEDING 3 PFD LIMIT	
WRO45CD60	4.817 PFD
WRO50CB60	3.225 PFD
WR225CD60	3.325 PFD
WR225DH60	3.302 PFD

3.8 DELAY PATH VERIFICATION

Delay Path Verification (DPV) is essentially an extension of design verification (DV). DPV involves the analysis necessary to assure that the chip will interact properly with the surrounding logic, as well as function internally as implemented by the physical design. These considerations enter into the design from the beginning, but become the most detailed just before and during physical design when much more accurate delay information is available. For the SCS481, this primarily involved checking that the propagation delays of the SCS481 met or exceeded those specified for the 54S481. Also included was the cursory manual analysis of the timing in the AP-101C and the effect of the design error correction circuitry.

In a normal chip design, this step in the methodology would involve extensive calculated delay simulation (Section 3.5.2.3) and timing analysis (Section 3.5.5). Since the SCS481 was a map-over of an existing chip, only a select few critical paths were delay simulated, and no timing analysis was done. Due to unfamiliarity with AP-101C design and unavailability of the BDL/S for the design, neither VMS nor TA would have been of much use for analyzing the interaction of the SCS481 with the surrounding logic. A manual timing analysis for the critical multiply instruction was deemed sufficient.

3.8.1 DELAY SIMULATION TIMING COMPARISON - 54S481 VS SCS481

The estimated and calculated critical path delays during various stages of the design are shown in Table 3.8-1. The TI specifications are taken from "the Bipolar Microcomputer Components Data Book" for the 54S481. (At the time of this writing, 54S481's were not available from TI. 54S481 specifications were used because they should represent the performance of the 74S481 at military temperatures).

The initial estimates turn out to be within +16% and -10% of the as wired delay calculations confirming the assumption that the estimate would be slightly pessimistic. The nominal delay measurements were all higher

Table 3.8-1 54S/SCS481 Critical Path Performance Comparison

KEY CRITICAL PATHS

PATH	TI SPECS		INITIAL* ESTIMATE (PESSIMISTIC)	DELAY SIMULATION*		
	TYP	MAX		NOMINAL	NOM	WC
MULTIPLY OP9 #	45NS	90NS	29.2NS	34.2NS	32.4NS	48.6NS
DIVIDE OP9 (FIX) a	45NS	90NS	38.6NS	44.4NS	35.5NS	53.3NS
A/B BUS TO X,Y	32NS	60NS	30.6NS	32.4NS	28.3NS	42.5NS
A/B BUS TO AG, LG	60NS	100NS	40.2NS	43.3NS	35.9NS	53.9NS
A/B BUS TO EQUAL	45NS	75NS	35.4NS	35.6NS	30.4NS	45.6NS
A/B BUS TO DATAOUT%	42NS	75NS	35.4NS	43.3NS	38.4NS	57.6NS
A/B BUS TO OVERFLOW	35NS	60NS	35.4NS	35.5NS	31.5NS	47.3NS
CIN TO COUT	30NS	50NS	16.2NS	19.3NS	17.8NS	26.7NS
OPBUS TO DATAOUT % \$	70NS	115NS	38.8NS	37.9NS	40.8NS	61.2NS

* OFF CHIP DRIVERS ARE ASSUMED TO HAVE A NOMINAL DELAY OF 5NS. THE ESTIMATE ASSUMES A WEIGHTED AVERAGE OF 1.6NS PER BLOCK

THIS PATH CONTAINS 3 OCD'S AND SOME INTERCHIP DELAY ASSUMED EQUAL TO ZERO

@ THIS PATH IS DETERMINED BY CIN PROPAGATION. THESE FIGURES ARE FOR ONE SCS481 ONLY

\$ THIS PATH IS FOR OP FORM 1 ONLY

% THESE TWO PATHS CONTAIN THE HIGH CAP NETS WORST CASE COMBINATION. THE DATA REFLECTS THE CAPACITANCE IMPACT OF THESE NETS

than the as wired delay calculations except for the OPBUS to DATAOUT delay, again corroborating the fact that the wiring capacitance default values yielded conservative delay calculations.

The nominal delay calculations (Section 3.5.2.2) and the as-wired calculated delay values (Section 3.5.2.3) are underneath the delay simulation heading. As explained, the nominal delay calculations take into account fan-in and fan-out and assume a reasonably high default wiring capacitance value. As a result, the nominal delays are more accurate than the initial estimate, but tend to be slightly pessimistic due to the capacitance default. The as-wired delays are as accurate as can be calculated by EDS taking into account the actual wiring capacitance. The worst case (WC) delay is the delay for 100°C junction.

The multiply OP9 critical path starts at the \uparrow clock transition. The path includes the propagation of the new least significant bit in the XWR from the LSP to the MSP, through the macro decode logic, and out the OP9 output. The inter-package delay was assumed to be zero and the two OCD's were assumed to be 5ns each. This is the critical path in the AP-101C which determined the multiply speed.

The divide OP9 critical path occurs in the Fix Remainder step (XId) in the signed divide OP Form. The propagation of this signal is actually dependent on the propagation of the \overline{CIN} to the MSP. This calculation assumes the \overline{CIN} is already available upon the \uparrow clock transition. The other paths measured are self explanatory.

The initial estimates turn out to be within +16% and -10% of the as wired delay calculations confirming the assumption that the estimate would be slightly pessimistic. The nominal delay measurements were all higher than the as wired delay calculations except for the OPBUS to DATAOUT delay, again corroborating the fact that the wiring capacitance default values yielded conservative delay calculations.

Comparing the as wired nominal and worst case delays for the SCS481 with the typical and maximum specifications for the 54S481, all the SCS481 propagation delays are less than those for the 54S481. The SCS481 nominal delays are on the average 26% faster than the 54S481, and the worst case delays average 38% faster.

Two caveats for this comparison are necessary. First, the worst case calculations for the SCS481's are for 100°C junction whereas the TI maximum should be for 125°C junction. Secondly, many of the SCS481 propagation delays are derived from just a few simulation examples selected at random, and therefore, may not be absolutely worst case propagation delay for the signals. The use of the Timing Analysis program (Section 3.5.5) would automatically figure worst case propagation delays.

With the values obtained from this analysis, no problems were anticipated in substituting the SCS481 for a 54S481, since all the 54S481 propagation delay specifications checked were met or exceeded. Even with the conservative latch speed design (Section 7.4), the MS399 technology was sufficiently faster than the TTL logic in the 'S481 so that no noticeable speed loss occurred.

3.8.2 PHYSICAL DESIGN CAPACITANCE CHECKS

DPV also requires a check on certain wiring and load capacitance values once the physical wiring has been accomplished. First, the wiring and load capacitance on the negative and positive phase of the clocks coming from the clock drivers must be manually checked to assure that the ratio of the positive phase wiring capacitance to the negative phase wiring capacitance is less than 2.0. This assures that skew caused by net capacitance will not allow the negative phase to arrive at the latch before the positive phase. Table 3.8-2 shows all the clock drivers and their respective total capacitance (pin 20 is negative phase, pin 60 is positive phase). All of the ratio's are less than 2.0 with the highest equal to 1.85, so no problem was anticipated.

Secondly, any nets with total capacitance above 3pf are flagged by the physical design programs and must be manually checked by the designer to assure that this will not adversely impact the performance of the device. Table 3.8-3 shows the four nets which were flagged during physical design. Figures 3.8-4, 3.8-5, and 3.8-6 show what areas of the logic these nets affect. The delay impact column of Table 3.8-3 indicates the improvement in performance which would accrue if the capacitance value were reduced to 3.0pf. Note that net WRO45CD60 is dependent on net WR225CD60 so that the potential delay impact for these two nets is the sum of their individual impacts. The potential delay impacts were small enough that no manual re-routing of wires was deemed necessary to reduce the capacitance.

3.8.3 AP-101C TIMING ANALYSIS

DPV also involves checking to see if the chip will function with the surrounding logic. Since the design of the AP-101C was completed elsewhere (IBM Owego, NY), and to keep design analysis and learning to a minimum, only the key critical multiply timing was analyzed. Figure 3.8-7 illustrates the functional block diagram of the execution unit for the AP-101C.

Table 3.8-2. Clock Driver Net Capacitances

Clock Driver		Total Capacitance (in pf)		Difference (60-20)	Ratio 60/20	Latches Driven
		Pin 60	Pin 20			
WR001	BA	1.85	1.29	0.56	1.43	PC, MC, XWR Sign (B Clock)
	BB	1.32	0.87	0.45	1.52	WR, XWR, R Fixed (B Clock)
	BC	0.92	0.80	0.13	1.16	A, B, Text Circuits (B Clock)
	CA	1.31	0.71	0.6	1.85	PC, MC, (C Clock)
	CB	1.57	0.89	0.68	1.76	WR, XWR (C Clock)
	CC	0.89	1.07	(-0.18)	0.83	R Fixed, XWR Sign, BP (C Clock)
	CD	0.58	0.52	0.06	1.11	Stack 0 (C Clock)
	CE	0.45	0.46	(-0.01)	0.98	Stack 1 (C Clock)
	CF	0.53	0.48	0.05	1.1	Stack 2 (C Clock)
	CG	0.65	0.46	0.19	1.41	Stack 3 (C Clock)
WR003	DA	0.56	0.52	0.04	1.08	A Latch (C Clock)
	DB	0.53	0.66	(-0.13)	0.80	B Latch (C Clock)
	DC	1.23	1.23	0	1.0	A, B, Test Ciruits (A Clock)
	DD	1.07	0.79	0.28	1.35	PC, MC (A Clock)
	DE	1.04	1.06	-(0.02)	0.98	WR, XWR (A Clock)
	DF	0.96	1.13	(-0.17)	0.85	R Fixed, XWR Sign, BP (A Clock)
	DG	0.86	0.72	0.14	1.19	Stack 0, Stack 1 (A Clock)
	DH	0.72	0.73	(-0.01)	0.99	Stack 2, Stack 3 (B Clock)
WR158	CA	0.31	0.27	0.04	1.15	BP (C Clock)
	CB	1.11	0.87	0.24	1.28	Stack 0, Stack 1 (B Clock)
	CC	0.48	0.39	0.09	1.23	Stack 2, Stack 3 (B Clock)

Table 3.8-3. Capacitance Delay Impact

NET	Name	Capacitance	Delay Impact	As Wired Delay Calculated
WR045CD60 *	SUM BUS BIT 3	4.81 PF	0.87NS \pm 50%	3.64NS F
WR050CB60	SUM BUS BIT 1	3.225 PF	0.2NS \pm 50%	5.31NS R
WR225CD60	LOGICAL OPER.	3.325 PF	0.28NS \pm 50%	4.79NS R
WR225DH60 *	RIGHT SHIFT-1	3.302 PF	0.15NS \pm 50%	2.89NS F

NOTE: Worst case impact would be WR225CD60 Delaying WR045CD60.
Total impact would be 1.15NS \pm 50%.

* Connected to high power blocks.

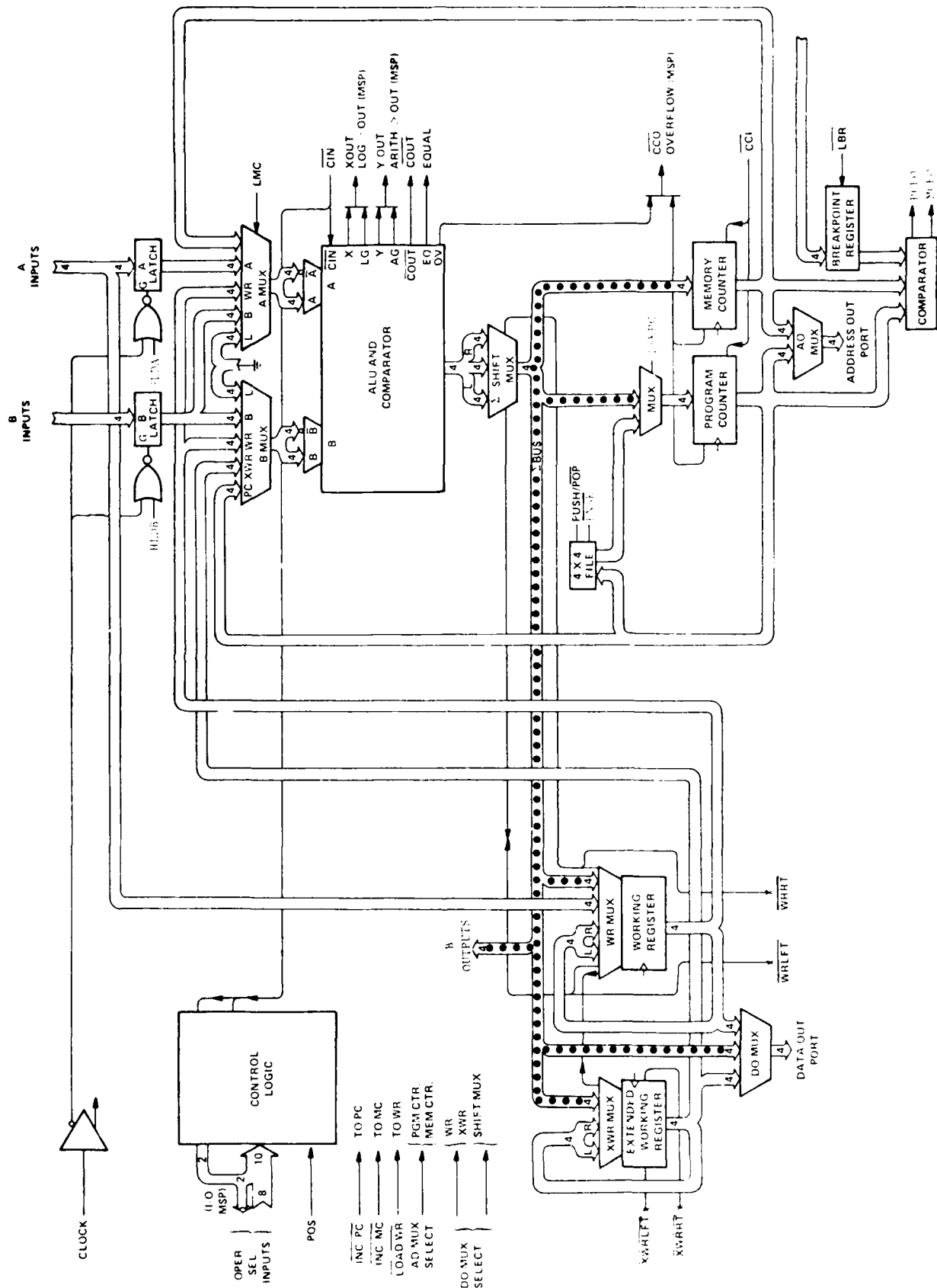


Figure 3.3.4 Advanced Delay Impact for Net WK045CD60 (0.87NS)

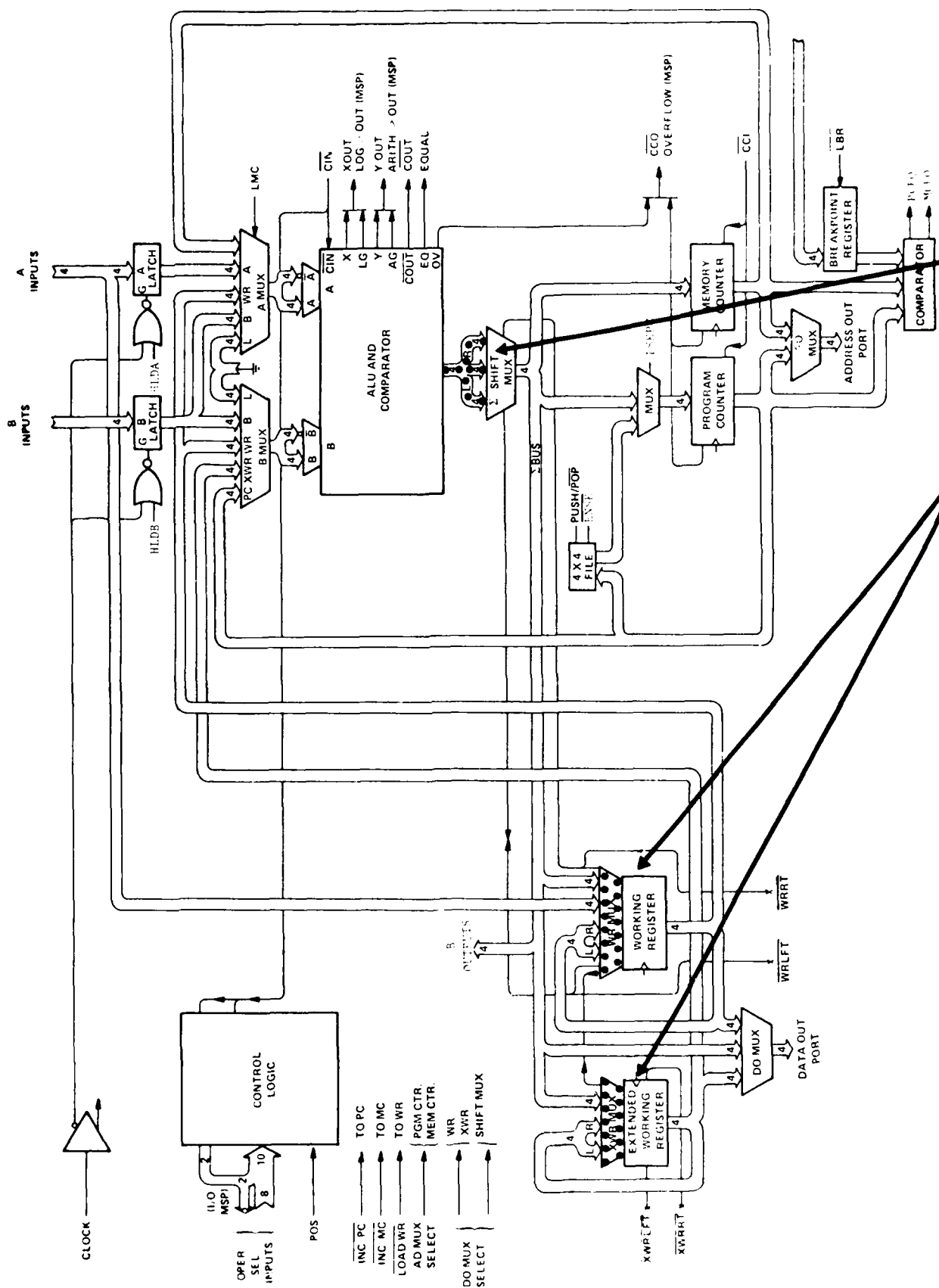
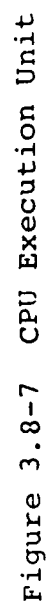


Figure 3.0-6 Capacitance Delay Impact For Nets WR250H60 (0.15NS) WR250D60 (0.28NS)

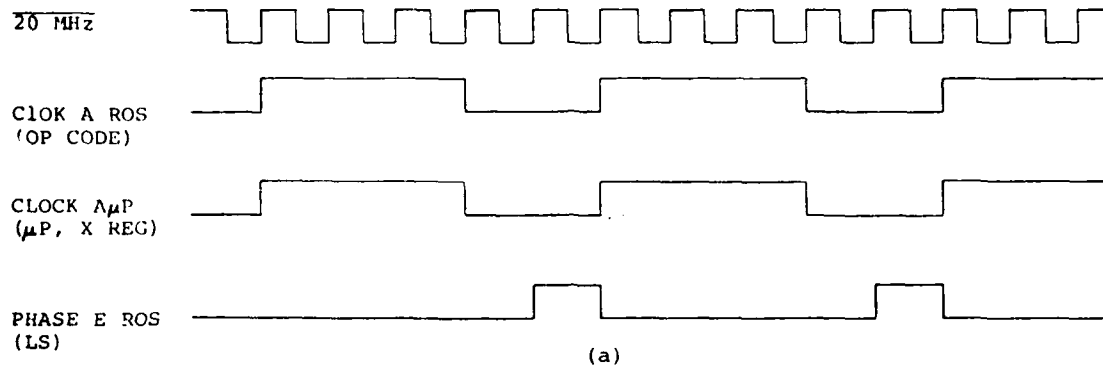


First the various clocking modes for the AP-101C had to be deciphered. Figure 3.8-8 shows the five basic clocking modes for the AP-101C. All clocks are derived from the 20 MHz master oscillator. CLOK A ROS loads a new microcode word on its I clock transition. Clock A mP feeds the '481, X register, and iteration counter clock inputs. The Phase E ROS clock causes data to be written into the local store RAM. CS 66 and 67 are the clock speed control bits in the micro instruction word. Figure 3.8-8(b) is the clocking mode used for high speed interactive operations like shift, multiply and divide. The first cycle is 200ns, followed by 150ns clock cycles until the iteration counter reaches a zero value.

The worst case timing analysis for the first cycle in the multiply instruction is shown in Figure 3.8-9. All signals are represented with a single line spreading out to double lines for data valid. The start of the sloping double line indicates the sum of the typical circuit delays and the end of the sloping lines indicates the sum of the maximum circuit delays. The data actually becomes valid somewhere in between.

AP-101C CLOCKING

250 NS NOT HIGH SPEED [CS66,67 = 00B]



200 150 NS HIGH SPEED [CS66,67 = 00B]

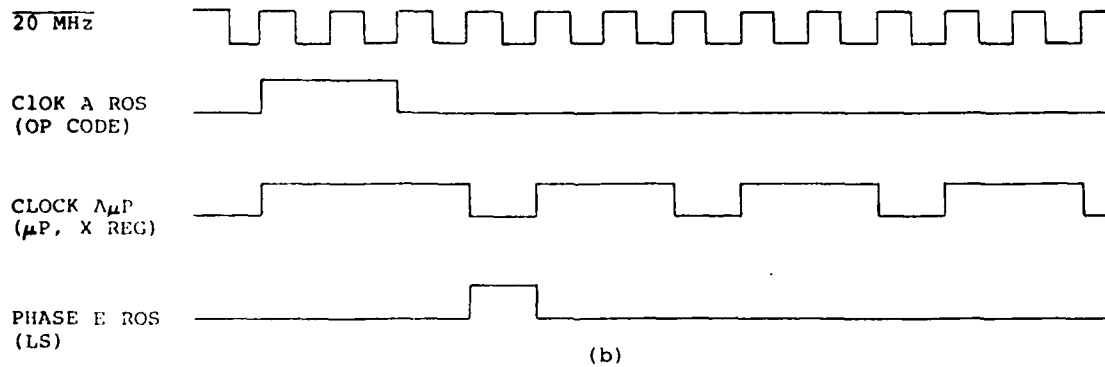
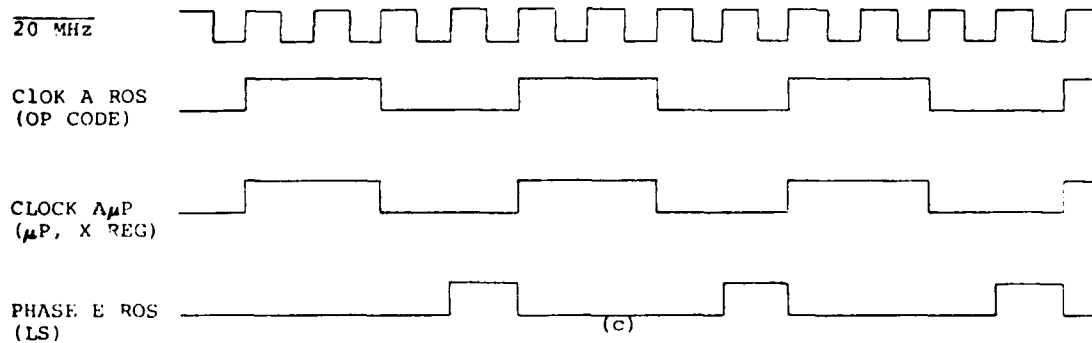


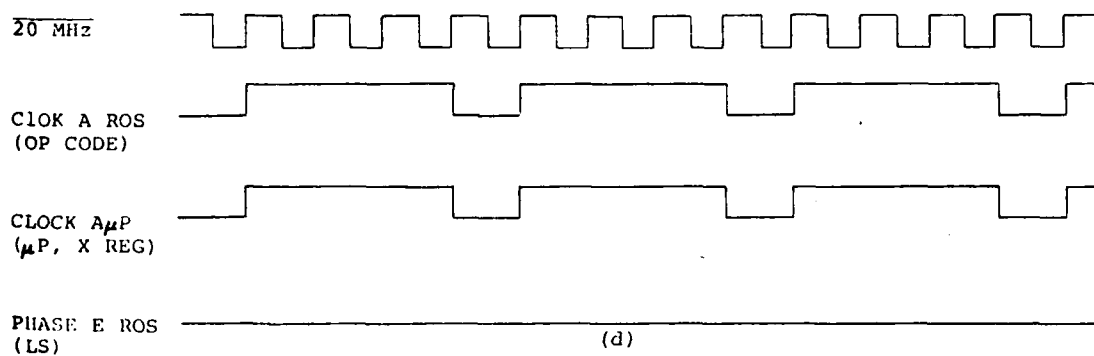
Figure 3.8-3

API01C CLOCKING

200 NS [CS66,67 = 01B]



200 NS [CS66,67 = 10B]



150 NS [CS66,67 = 11B]

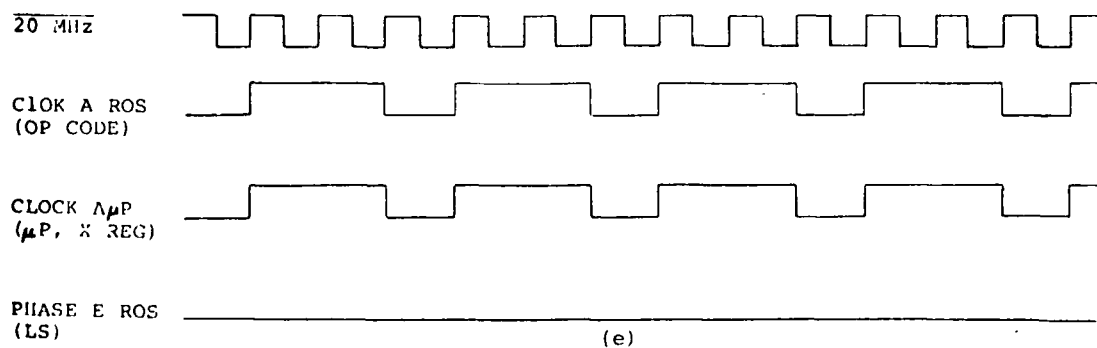
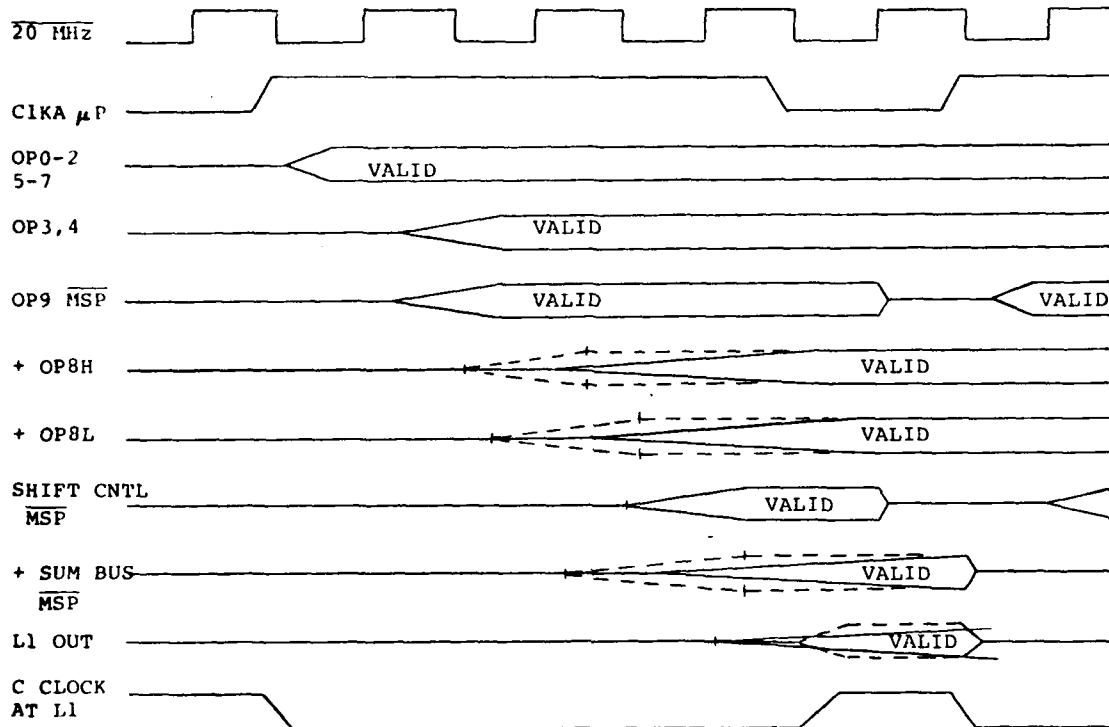


Figure 3.3.3 (continued)

AP-101C MULTIPLY TIMING (Worst Case)* (1st Cycle)



* WORST CASE IMPLIES THAT CS 19,20,21 HAVE CHANGED FROM PREVIOUS VALUE

+ THIS TIMING REFLECTS DATA WHICH TI HAS NO EXPLICIT SPECIFICATIONS FOR. THE DOTTED LINES ARE FOR AN SCS 481. THE SOLID REPRESENTS TI 'S481 USING CLOCK TO OP8 TIMING SPEC. THE REAL TIMING IS SOMEWHERE IN BETWEEN.

Figure 3.8-9

The two OP8 propagation delays involve a logic path in the 54S481 for which no explicit specification is given. This delay is from a valid OP Bus to a valid OP8. The dotted lines represent the SCS481 delay for this signal; the solid lines indicate this delay using the clock to OP8 timing specification for the 54S481. The actual delay is probably in between these two.

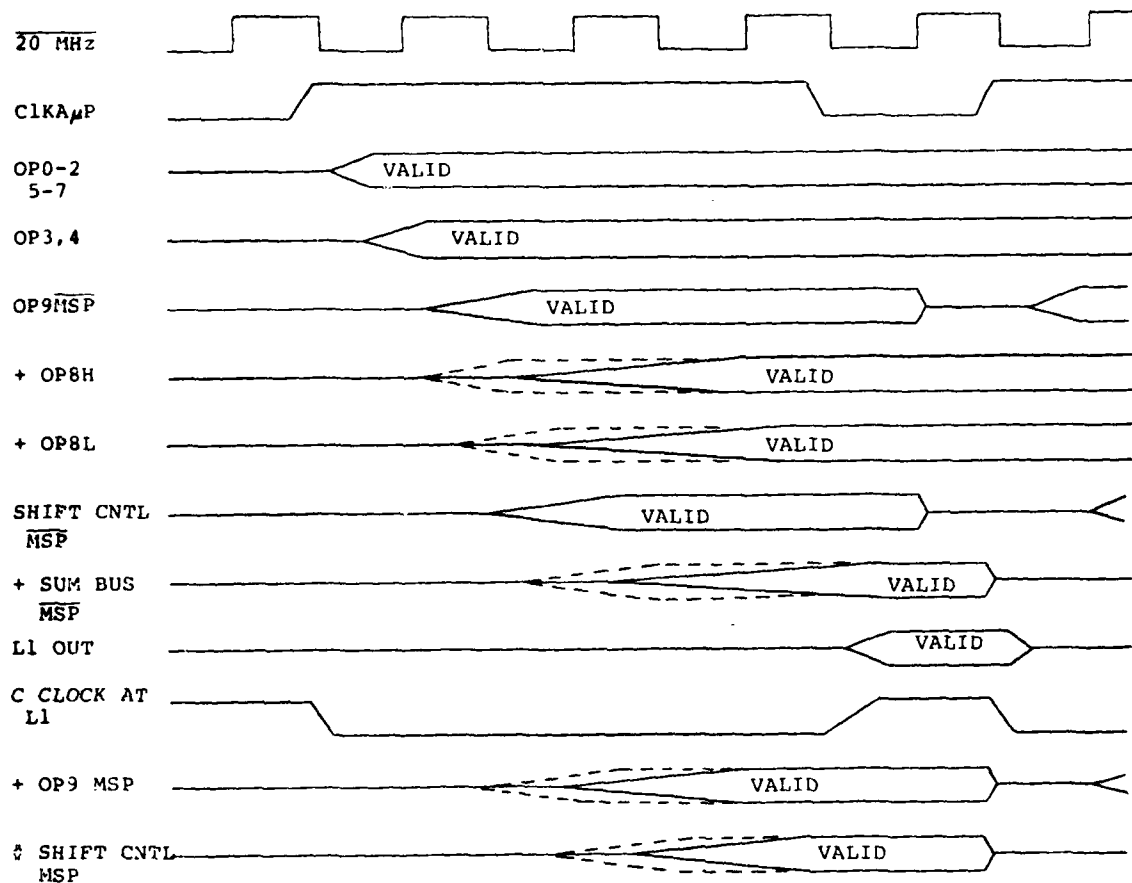
The last four signals are internal signals to the SCS481. The L1 output signal represents the output of the L1 latches for the WR and XWR. This data must be valid before the falling edge of the L1 C Clock for proper operation. The shift control and Sum Bus must be valid for the L1 output to be valid.

For the worst case timing shown, a fair chance exists that the wrong data will get latched into the WR or XWR. This worse case timing assumes that all microinstruction bits change on the initial rising edge of the Clok A. Microinstruction word bits CS 19, 20, and 21 are the critical bits. These bits control the input MUX to the B-Bus. Since the B-Bus is not used in the multiply, these bits do not change in the actual implementation of the multiply.

Figure 3.8-10 shows the multiply timing for the case in which CS 19, 20, and 21 do not change on the initial edge of Clok A. In this case, the L1 data outputs are valid at least 20ns before the falling edge of the C Clock. Two signals internal to the 54S481 in the MSP of AP-101C are hypothetically shown. The times shown are based on 1.35 times the SCS481 internal timings for the same paths. The shift control line for the MSP must be valid before the rising edge of Clock A mP for proper operation. This signal is valid more than 45ns early worst case so no problems were anticipated for the first cycle of multiply.

Figure 3.8-11 shows the timing for the remaining fifteen multiply cycles. For the SCS481, the L1 outputs are valid as soon as the C Clock rises with more than 75ns of slack in the Sum Bus valid data path. The

AP101C MULTIPLY TIMING ACTUAL PRACTICE* (1st Cycle)



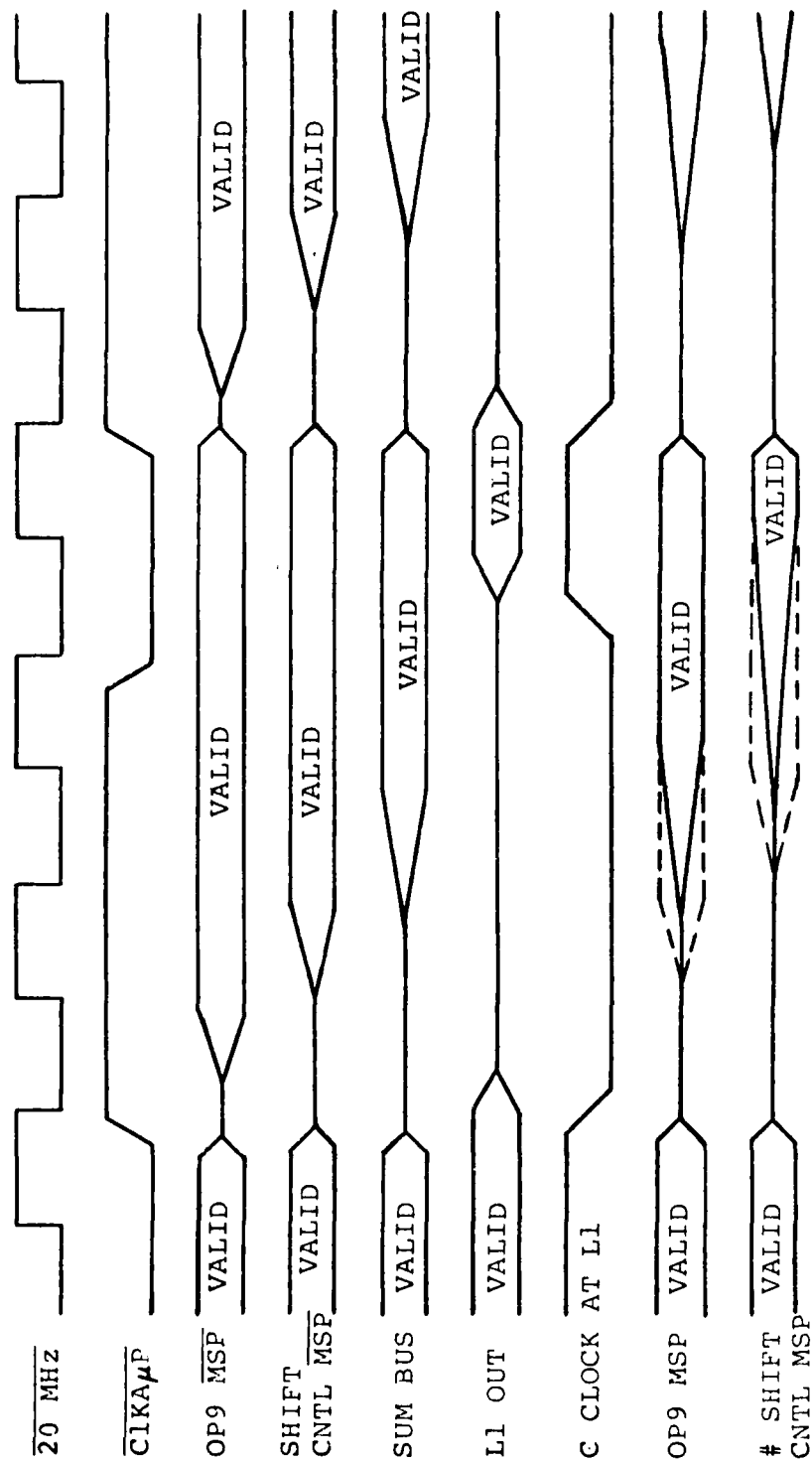
* ACTUAL PRACTICE IMPLIES THAT CS19,20,21 HAVE NOT CHANGED FROM PREVIOUS MICRO INSTRUCTION REFLECTING ACTUAL PRACTICE.

+ THIS TIMING REFLECTS DATA FOR WHICH TI HAS NO EXPLICIT SPECIFICATION. THE DOTTED LINES ARE TIMINGS FOR AN SCS 481. THE SOLID LINES REPRESENT TI's 481 TIMING USING THEIR "CLOCK TO OP8" TIMING SPEC. THE REAL TIMING IS SOMEWHERE IN BETWEEN.

THIS TIMING IS AN ESTIMATE BASED ON 1.35 TIMES THE SCS 481 TIMING.

Figure 3.8-10

AP-101C MULTIPLY TIMING - REMAINING CYCLES



THIS TIMING IS AN ESTIMATE BASED ON 1.35 TIMES THE SCS 481 TIMING.

Figure 3.8-11

shift control in the MSP 54S481 has only 15ns to spare in the maximum delay case. Since all tests were to be at room temperature, no problems were anticipated for any of the multiply cycles.

3.8.4 EFFECTS OF DESIGN ERROR CORRECTION CIRCUITS

In the process of design, two logic errors were introduced which were not caught during design verification. The counter carry in (\overline{CCI}) in the least significant slice functions in an inverted fashion from the TI \overline{CCI} signal; i.e., a low causes a double increment and a high causes a single increment. All four shift bits, $WRRT$, $WRLFT$, $XWRRT$, and $XWRLFT$, are inverted for both input and output when compared to the TI signals. External circuitry is required to correct these errors. The timing effects of these circuits are analyzed in the following paragraphs.

The effect of the inverter on CCI which is necessary for the LSP to correctly increment the PC and MC is virtually negligible. With the extra speed inherent in the SCS481, adding an extra inverter delay will not greatly affect the overall incrementing speed. In SCS481 stand alone designs (i.e., when a direct substitution of a 74S481 is not desired), the inverter is unnecessary.

The effect of the shift bit inversion circuitry is more significant. The multiply instruction is affected by this circuitry because the least significant bit of the XWR must pass out the $XWRRT$ pin and through this circuit before reaching the MSP 74S481. Once this bit reaches the MSP, then that 74S481 can decide whether to add and shift or just shift. As pointed out in the previous section, only about 15ns of slack exist in this signal path, so this circuitry must operate in that period of time.

The initial circuitry used to correct these errors is shown in Figure 3.8-12. The circuit uses a pull-up resistor pair on the SCS481 open collector shift bit and a pull-down resistor for the three-state 74S481 shift bit. Whichever bit creates an active level shift of the opposite polarity first

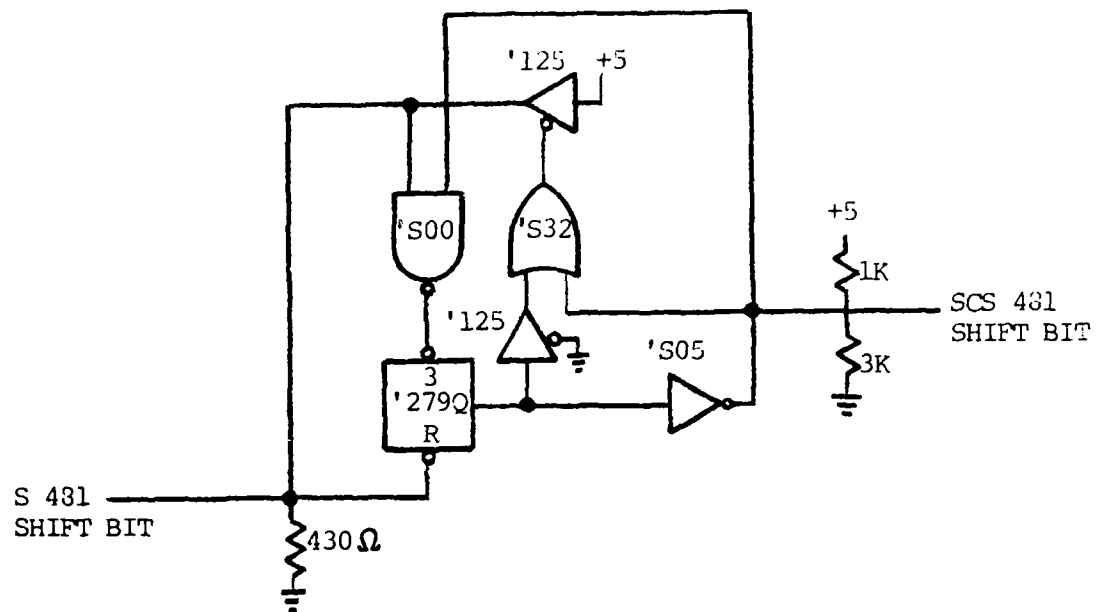


Figure 3.3-12 Initial Shift Bit Inversion Correction Circuitry

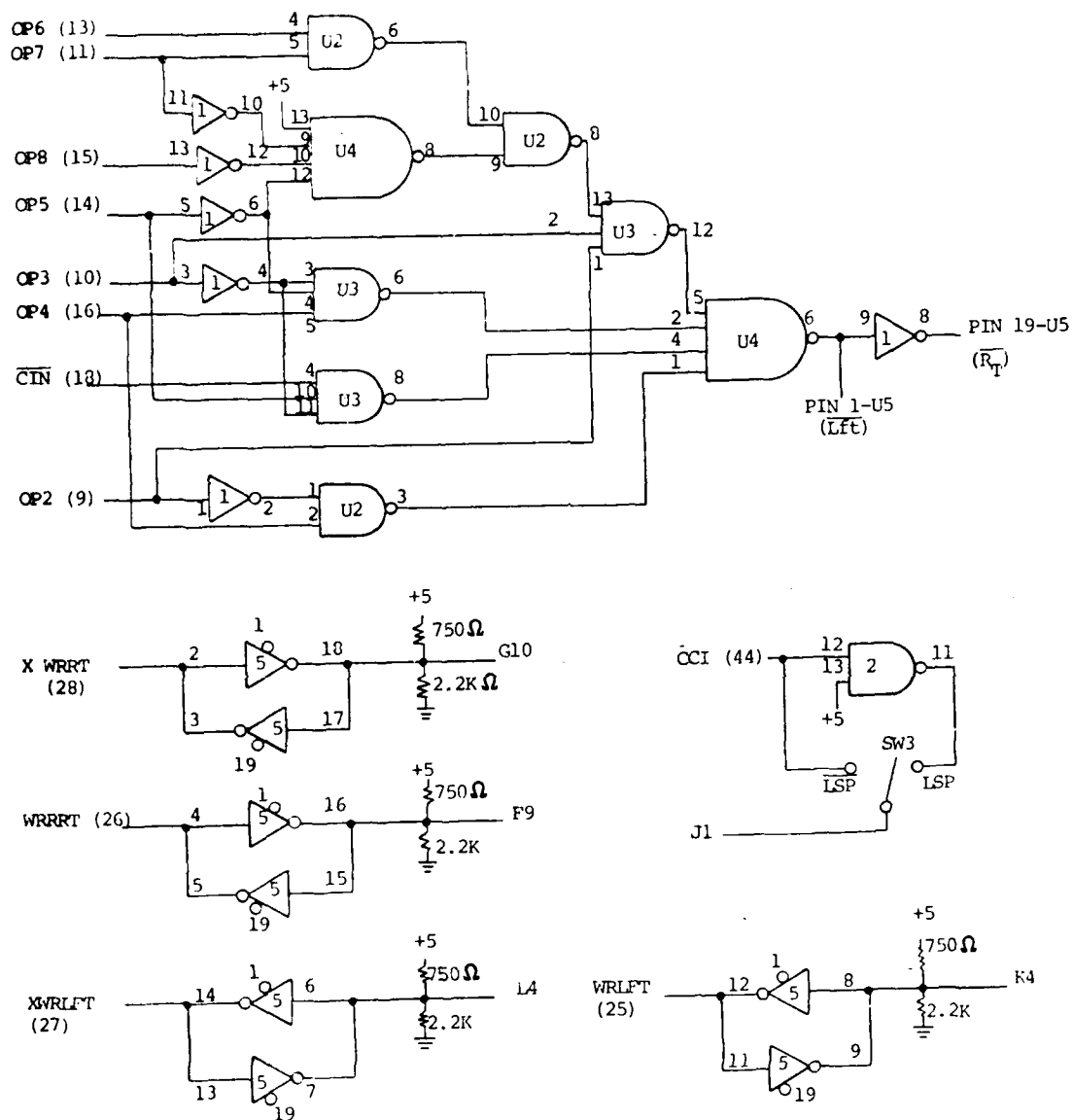


Figure 3.8-13 Circuits Required To Correct SCS 481 Logic Inversions

(i.e., SCS481 bit going low or the 74S481 bit forced high) takes control of the circuit and forces the other bit to the opposite polarity. Logic analyzer results showed that from the rising edge of the clock, this circuit was able to propagate a zero in the SCS481 as a one to the 74S481 in less than 30ns, well within the TI specification of 40ns.

In transferring a bit of the opposite polarity, the SCS481 bit must float up and activate the circuitry which disables the 54/74125 output; then the 74S481 bit must float down. The logic analyzer showed this to take almost 100ns from the rising clock edge. This circuit would not work at 20 MHz, but did function properly at 18.3 MHz. This meant that the propagation delay needed to be decreased by about 14ns (18.3 MHz has a period of 54.6ns. Since the multiply cycle consists of three periods, the extra time per cycle at 18.3 MHz is 3 times 4.6 or 13.8ns).

The circuitry which finally worked at 20 MHz is repeated in Figure 3.8-13. The nand gates and inverters decode portions of the OP Code to determine whether a left or right shift is to be performed. These signals then control which three-state inverter is enabled between the two shift bits. This allows for an active drive of both shift bits. The logic analyzer determined that this circuit operated in less than 40ns from the rising clock edge in all cases. Again, this circuitry would be unnecessary in a stand alone SCS481 design.

3.9 TEST PATTERN GENERATION (LSSD)

The primary intent in the development of LSSD was to provide design guidelines which would allow automatic test pattern generation for any design regardless of complexity. The following descriptions of test generation for the SCS481 demonstrate the automatic test features of LSSD. As will be explained, LSSD has the ability to relieve the designer from the drudgery of generating manual test patterns for his design by delegating this responsibility to the computer. A complete description of the EDS test generation methodology and the test generators which are available follows.

3.9.1 TEST GENERATION DEFINITIONS

Before describing test generation the following terms should be defined.

Stuck fault or Fault - is the condition which occurs due to some manufacturing process error which causes either the input or output pin of a logic gate to remain at either a logical 1 or 0 value independent of the values input to the gate. The majority of process errors can be properly represented as a stuck fault. Test generation programs attempt to generate patterns which will detect all possible stuck faults by forcing each gate pin to the opposite value from the fault and then creating a "sensitized path" from primary inputs or SRL's to primary outputs or SRL's. Input pin stuck faults are only physically distinguishable from an output pin stuck fault in the non-controlling position. The equation for the number of stuck faults per elementary logic gate is $N+2$ where $N \geq 1$ and equals the number of input pins. For example, NOR gate 1 in Figure 3.9.1-1 has five potential stuck faults: Pins A, B and C stuck at 0 and pin 0 stuck at 0 and 1.

Reduced Fault - is a stuck fault which is physically indistinguishable from another stuck fault. For example, in Figure 3.9.1-1 pin 0 of NOR gate 1 stuck at 0 (s-a-0) is indistinguishable from pin A of NOR gate 2 s-a-0 and, therefore, is a reduced fault. In contrast, Pin 0 of gate 2 s-a-0 is different from both pin B of gate 3 and pin A of gate 4 s-a-0 and is not reduced.

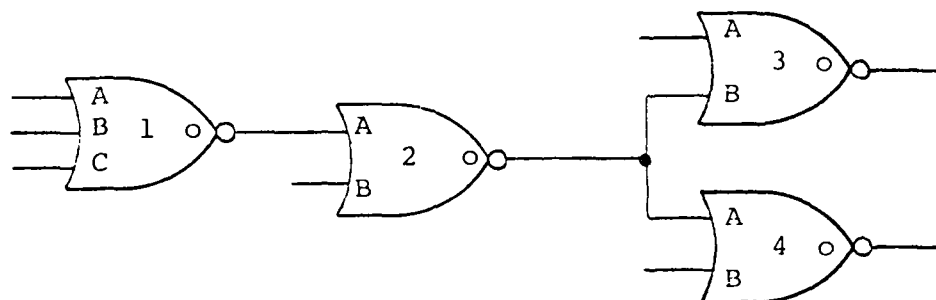
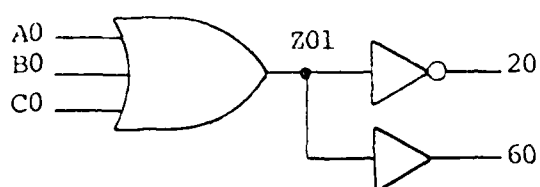
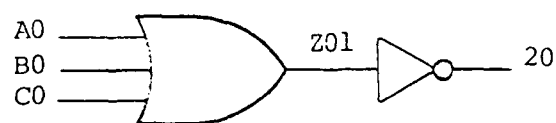


Figure 3.9.1-1



a) TEST MODEL FOR CEB0



b) TEST MODEL FOR CEB0 WITH PIN 60 DROPPED

Figure 3.9.1-2

Possibly Detected Fault - is a fault whose detection is dependent upon an undetermined value at a primary output pin during fault simulation. Two common causes can be an undetermined state of a latch upon power up or if the fault can cause oscillation or a race condition which the simulator cannot evaluate.

Untestable Fault - a fault for which no pattern can be generated to detect its existence. In LSSD designs untestable faults are almost universally caused by redundant logic design and, therefore, are sometimes referred to as redundant faults.

Net Testable Faults - is equal to the total number of faults minus the number of reduced and/or redundant faults. Figure 3.9.1-2 shows the test model for book type CEBØ. This model is able to represent all physical DC faults excluding shorts, by using a DC stuck fault model. Figure 3.9.1-2 (b) shows what happens to this test model when one of the outputs is unused. Net Z01 is no longer necessary to represent all physical DC faults using a stuck fault model; however, it still appears in the test model. The two stuck faults for this net are flagged as reduced faults. Many of the reduced faults in an MS399 test model are of this nature. Because of these psuedo reduced faults and the fact that test generators do not try to generate patterns for reduced faults, both reduced and redundant faults are subtracted from the total number of faults to yield net testable faults.

Untested Faults - a fault for which no pattern has been generated. It is not necessarily untestable.

Good Machine - Represents the behavior of the design with no stuck faults; i.e., a good chip. Only one good machine exists.

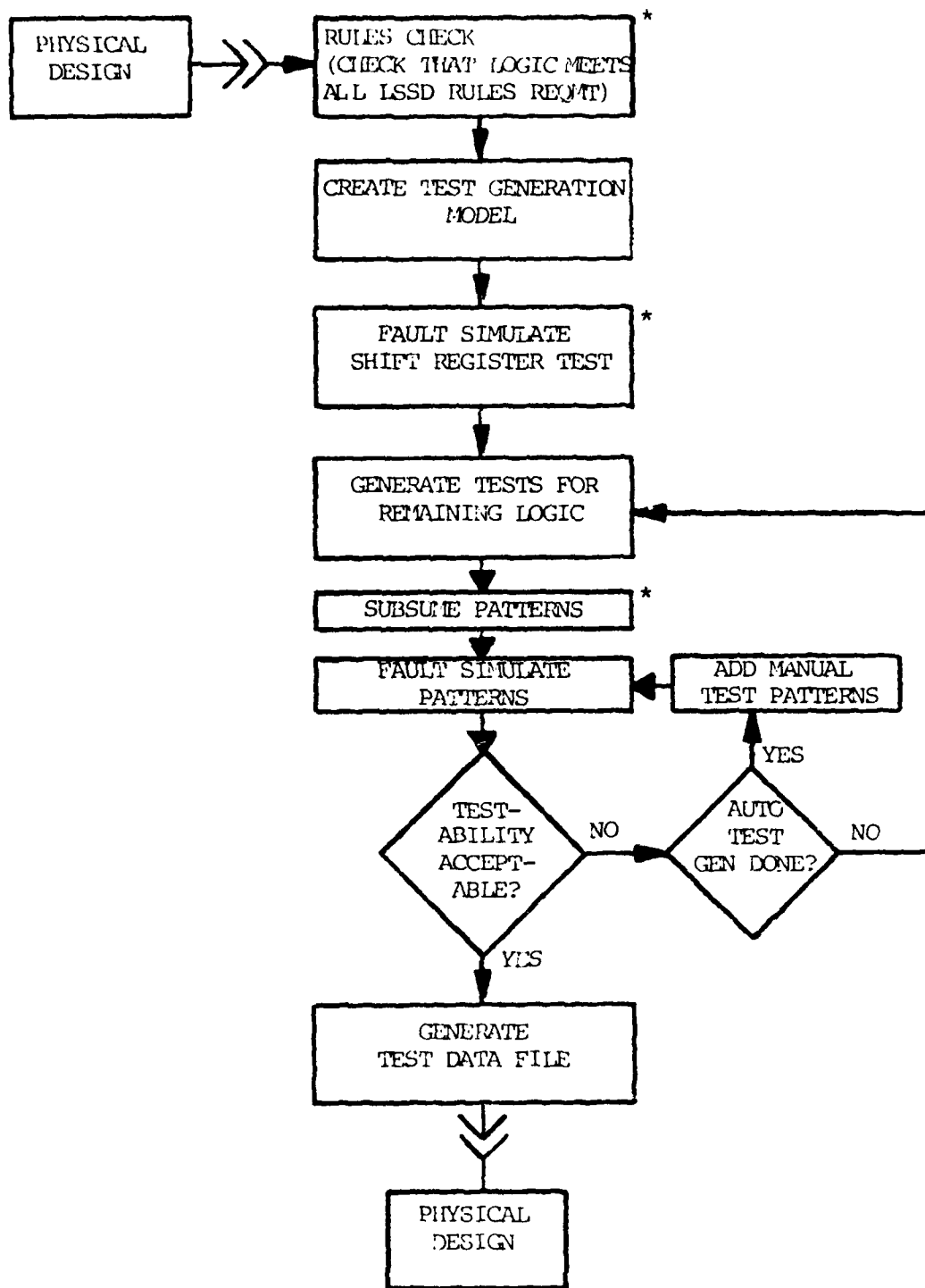
Fault Machine - Represents the behavior of the design with one stuck fault. The number of fault machines equals the number of potential stuck faults in the design.

3.9.2 EDS TEST GENERATION METHODOLOGY

Several steps are involved in the test generation process. Figure 3.9.2-1 shows the test generation methodology flow diagram for EDS. In general test model and fault dictionary must be generated first. This step takes note of any SRL's and their respective clocks and scan paths. An analysis of the circuit is performed to determine the number of reduced faults which are in the logic. After this analysis, the system is then ready for either automatic or manual test generation.

For automatic test generation, four different test generators with various options are available to choose from. These will be explained later in more detail. The procedure is to select one of these test generators with the appropriate options and run it. Figure 3.9.2-2 shows the approach taken by the non-LSSD test generators to generate patterns to detect faults. An untested fault is selected from the dictionary and the generator attempts to develop a pattern which will detect that fault. If a pattern can be generated, it is then fault simulated to determine what faults were detected by that pattern and they are "checked off" in the fault dictionary as tested. All fault machines flagged as tested are then no longer eligible for test generation or fault simulation. Any patterns which do not detect any faults are discarded except when the pattern is part of a sequence of patterns needed to test an embedded sequential fault. The test generator then continues until all faults are tested or tried or one of the options causes it to stop.

After the test generator has stopped, a percent tested figure and the number of patterns generated are printed out. By examining these figures the decision can be made whether the test generator was effective in detecting faults. If the results seem reasonably effective, those results are then saved so that the next test generator will not try to test faults detected by this test generator. The option not to save the results allows for the elimination of ineffectual pattern sets; therefore, conserving valuable tester time and keeping test pattern volume low.



TEST GENERATION

* INDICATE STEPS TAKEN FOR LSSD PARTS ONLY

Figure 3.9.2-1

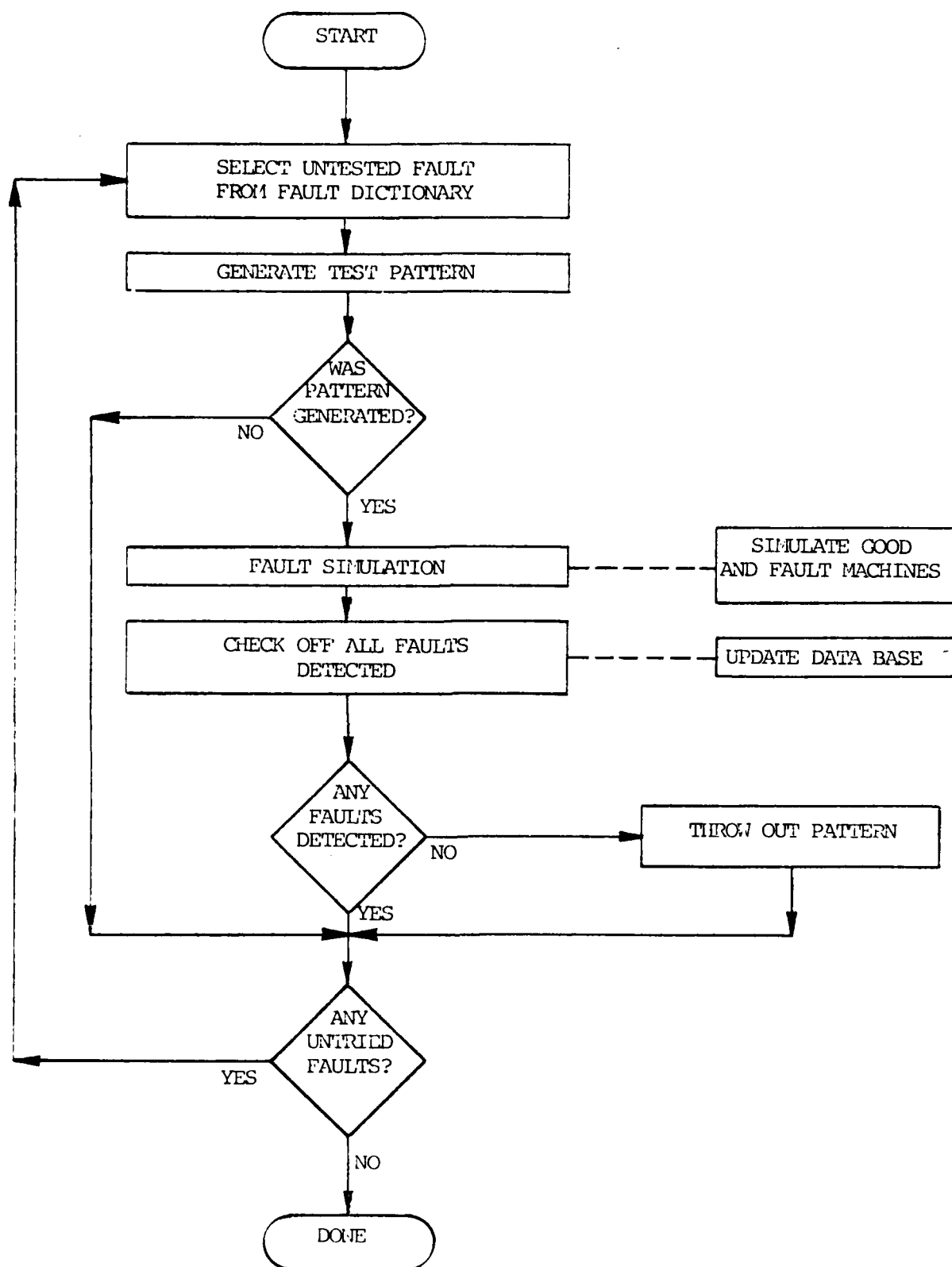


Figure 3.9.2-2 Non-LSSD Auto Test Generator Approach

Whether the results are saved or not, another test generator is selected or the options are changed on the previous test generator and the whole procedure is repeated. This iterative process is repeated until the desired test percentage is achieved or supplemental manual patterns must be generated. The approach is the same for manual patterns except the computer does not generate any patterns.

3.9.3 TEST GENERATORS

Four basic test generators are presently available to the EDS user: RANDGEN, FAULTGEN, PODEM, and SOFTG. Each of these have some important options which are explained below.

3.9.3.1 RANDGEN

As the name implies, this is a random test pattern generator. Random patterns are applied to the primary inputs; and, for LSSD, they are also applied to the scan string. In many cases this test generator has the ability to detect a lot of faults quickly; however, it can tend to be ineffectual for high fan-in circuits like PLA's or in non-LSSD circuits. The "time slice" option causes the test generator to stop after no new faults have been detected within a certain period of time.

3.9.3.2 FAULTGEN

FAULTGEN is designed to handle combinational logic. Given enough time and memory space, FAULTGEN can detect 100% of all faults in a non-redundant combinational circuit. It also has the capabilities to use the scan string in an LSSD design. In addition to the time slice option just described, three other options are frequently used: pattern slice, maximum decision number, and reverse. "Pattern slice" is similar to time slice except the test generator is stopped if new patterns cannot be generated after a certain period of time. The maximum number of times that decisions should be remade in detecting one fault is set by the "maximum decision number". The "reverse" option allows FAULTGEN to start at the end of the fault dictionary and work towards the beginning.

3.9.3.3 PODEM

PODEM is an LSSD test generator. PODEM has the additional feature that it will flag untestable faults in the logic; which, for LSSD almost always turn out to be redundancies. The same four options available for FAULTGEN are offered for PODEM.

3.9.3.4 SOFTG

SOFTG is designed to handle non-LSSD logic. It utilizes a path sensitizing technique and a built-in reward/penalty system which causes it to retry inputs that have allowed it to detect faults and avoid those which have been less useful. Iterative application of SOFTG is often successful. The four options previously described are available for SOFTG.

3.9.4 LSSD TEST TECHNIQUES

LSSD test methodology differs in some very fundamental ways from the non-LSSD version just described. These differences are indicated by an asterisk in Figure 3.9.2-1. First, before the test model is built, an LSSD rules checking program is run to assure rules conformance. Once rules checking is passed and the test model generated, a Shift Register (SR) test program, using information from the rules checking program and the test model, creates the following set of test patterns:

- a) scan clocks all on
- b) scan in set to 0,1,0; measure scan out.
- c) shift a 0,1,0 pattern through the shift string and measure expected output responses.

This SR test is sufficient to detect all faults in the SRL's except any data and system clock faults in the L1 latch. Those faults can then be treated as combinational faults with the SRL as a primary output. Depending upon the number of SRL's and the complexity of the logic, the SR test will usually detect 40% to 60% of the faults in the design in a relatively short set of patterns which are easily and universally generated.

From this point, test generation can begin in the iterative mode described in the test generation methodology. However, the LSSD test generators take a slightly different approach as shown in Figure 3.9.4-1. The subsume step examines the patterns generated after each test generation run to determine what patterns can be combined without affecting fault coverage. Since patterns are generated with one specific fault in mind, each pattern usually contains "don't care" conditions. Subsumption allows patterns which differ only by the other's respective "don't care" conditions to be applied simultaneously. Because LSSD reduces a circuit to a combinational level, combining patterns in this fashion is perfectly safe; whereas, a non-LSSD design cannot guarantee that pattern subsuming will achieve the same results. Subsuming saves fault simulation and tester time, as well as, test pattern volume.

Additionally, unless the circuit is extremely large and complex, LSSD test generation is so straight forward that the common procedure is to combine two or three test generators and submit them sequentially. Rarely does one have to examine test coverage after each step, decide whether to save the results, and then determine what new test generator options are required. Almost invariably, a judiciously selected set of two or three test generators will yield greater than 95% test coverage and will often result in 98% to 99% of the faults detected. This allows all the steps from rules check to test generation to be submitted sequentially and run overnight with little or no manual intervention.

3.9.5 LSSD TEST GENERATION PASS #1

As explained in Section 3.5.7, a preliminary testability analysis is usually completed before Physical Design by performing automatic test generation. LSSD test generation pass #1 was this testability analysis for the SCS481. Table 3.9.5-1 column 1 details the results from this run. In less than thirty-two 370/168 CPU minutes, 98% of the stuck faults were tested using 1761 patterns (LSSD pattern counts consider the scan in and the scan out each as one pattern).

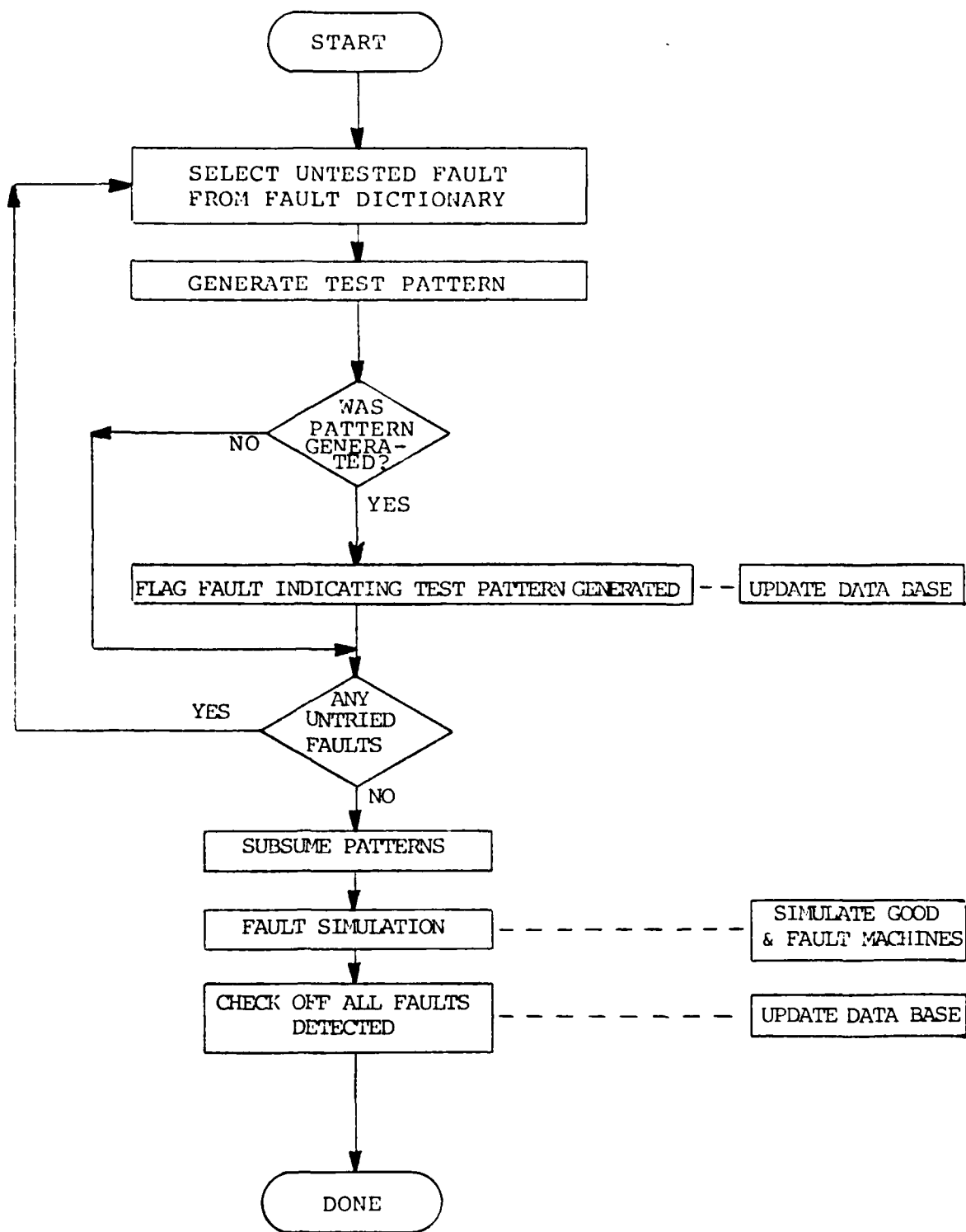


Figure 3.9.4-1 LSSD Auto Test Generator Approach

Table 3.9.5-1. LSSD Auto Test Generation Comparison

	LSSD Pass #1 W/O Test Circuits	LSSD Pass #2 W/ Test Circuits	
NUMBER OF SRL's	46	47	
NUMBER OF FAULTS	6723	7046	
NUMBER OF FAULTS DETECTED	6590	6875	
NUMBER OF FAULTS UNTESTED	133	171	
SRTEST DETECTED / CPU MIN	55.6 / 6:37	54.5 / 8:24	
RANDGEN DETECTED / CPU MIN	77.6 / 9:39	67.1 / 15:19	
PODEM DETECTED / CPU MIN	97.8 / 14:15	92.5 / 25.59	
FAULTGEN DETECTED / CPU MIN	98.0 / 1:18	97.5 / 6:21	
TOTAL DETECTED / CPU MIN	98.0 / 31:49	97.5 / 56:03	
NUMBER OF PATTERNS	1761	2275	
			Difference
MAX. DEC. REACHED PODEM	58	468	410
NO. OF FAULTS DETECTED PODEM	6577	6524	53
NO. OF TEST CIRCUIT FAULTS	-	323	-
NET FAULTS DETECTED PODEM	6577	6201	376

RANDGEN and the SR test are essentially non-algorithmic test generators whose CPU time is primarily spent doing fault simulation. These two test generators were able to achieve a 77.6% test coverage in just 16 minutes and 16 seconds. The remaining CPU time was consumed by the algorithmic test generators, PODEM and FAULTGEN, which use CPU time for test pattern generation as well as fault simulation.

The important thing to note is that once the design passed the rules check, the test model build and the four test generators were simply submitted in a serial queue in the morning and the results were available that afternoon with no manual intervention. This implies that test generation can be performed on an LSSD design without any explicit knowledge of the design or digital testing theory.

3.9.6 LSSD TEST GENERATION PASS #2

After the successful conclusion of LSSD test generation pass #1, the test circuits described in Section 3.1.5 were added to the design. Physical design was then successfully completed on the design and a final test generation pass was performed. The results from pass #2 are shown in Table 3.9.5-1 column 2. In about fifty-six 370/168 CPU minutes, 97.5% of the faults were tested using 2275 patterns.

In this case, RANDGEN and the SR test achieved 67.1% coverage in 23 minutes and 43 seconds. The remaining 32 minutes and 20 seconds were used by PODEM and FAULTGEN. The reasons for the slightly reduced coverage with increased CPU time for pass #2 are explained in the next section.

This pass of test generation was executed by the Design Automation department in charge of physical design. All the EDS job steps from expansion to test generation were serially submitted and executed overnight with no manual intervention or knowledge of the design.

3.9.7 LSSD TEST GENERATION COMPARISON

Even though pass #2 had a decrease in coverage and an increase in CPU time when compared to pass #1, the test circuits did not really decrease the testability of the design. In fact, the test circuit faults were completely tested. The main culprit appears to have been the extra latch added into the ring oscillator for LSSD rules conformance (see Figure 3.1.5-1). Detailed examination of Table 3.9.5-1 explains how this latch affected the test generators.

For pass #2 the SR test coverage was slightly decreased with a corresponding increase in CPU time. The time increase is due to the addition of the ring oscillator latch since additional patterns were necessary to complete the shift tests. The decrease in coverage probably reflects some additional test circuit faults which were not tested in the SR test.

RANDGEN exhibits the most significant difference between pass #1 and pass #2. The 10.5% coverage drop is linked to the modus operandi of RANDGEN. A random word is generated and scanned into the shift string. The remaining bits of the random word are then applied to the primary input pins. Even though the same seed was used to generate the random words, the addition of the latch to the end of the shift string caused the random patterns applied to the primary inputs to be "shifted" one bit. This fact resulted in the lower coverage and increased CPU time. Essentially, RANDGEN was not as "lucky" with this new set of primary input patterns.

The coverage decrease for RANDGEN required the algorithmic test generators, PODEM and FAULTGEN, to test an additional 812 faults. Looking at the PODEM statistics at the bottom of Table 3.9.5-1, one discovers that PODEM, after adjusting for the number of test circuit faults, actually detected 376 less faults in pass #2 than in pass #1. Part of the reason for this decrease can be found by examining the number of times the maximum decision count was reached. Pass #2 reached the maximum decision count 410 times more than in Pass #1. This would tend to indicate that the faults left untested by RANDGEN Pass #2 were more difficult for PODEM to test.

Past experience with PODEM and FAULTGEN have found that their CPU run time is proportional to the square of the fault count. For pass #1, PODEM and FAULTGEN were confronted with 1506 untested faults, compared to 2318 untested faults for pass #2. Using the squared relationship, these test generators should take 1.37 times longer to generate tests for pass #2 than they did for pass #1. PODEM and FAULTGEN actually took twice as long for pass #2. The added number of faults left-over from RANDGEN and the apparent additional difficulty associated with these faults account for the increase in CPU time and the 1/2 percent decrease in test coverage for LSSD test generation pass #2.

Another important fact to remember is that the remaining untested faults contained redundant and reduced faults. Once these faults have been subtracted, only 6 net untested faults exist for pass #1 and only 36 net untested faults remain for pass #2. Of these faults all six in pass #1 and four in pass #2 are due to the limitation that a skewed loading of the scan string was not possible in EDS (Section 3.1.3.4). This means that in less than thirty-two minutes, pass #1 essentially detected all the faults which could have been detected for that release of EDS. Pass #2 left only thirty-two faults which it could have tested, but were not.

3.9.8 FAULT SIMULATION OF FUNCTIONAL PATTERNS

One possible way to generate test patterns for a design is to use the functional test patterns developed for design verification. In order to do this, the functional patterns need to be fault simulated to determine the actual fault coverage. A methodology for doing this is illustrated in Figure 3.9.8-1. This methodology was used for fault simulation of a few of the functional pattern data sets.

The eight BDL/C data sets actually fault simulated are described in Table 3.9.8-2. All of these pattern sets except for Flush were simulated during design verification. Flush was written specifically to test the scan path, since none of the functional pattern sets exercise it. Flush merely turns on the A and B clocks and flushes a one and then a zero down the scan path.

MANUAL TEST GENERATION METHODOLOGY

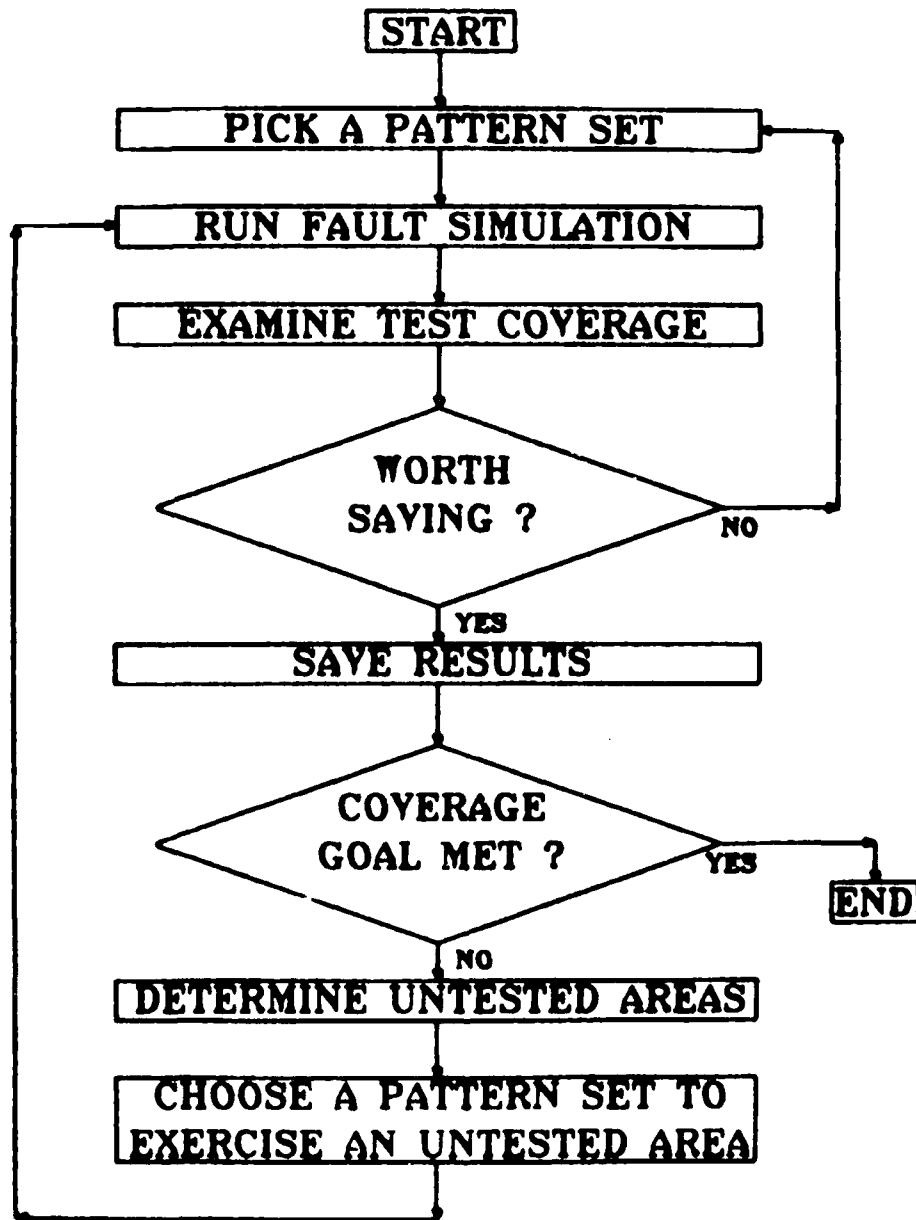


Figure 3.9.8-1

Table 3.9.8-2. BDL/C Data Sets for Manual Test Generation

BDL/C	Areas of Logic Exercised
ALU1 & ALU2	Load WR from ABUS, A & B MUXES all combinations, ALU ADDITION, SUMBUS to PC,MC,WR,XWR, and NOOP.
TEST/ENHANCE2	Test circuits, stack, and PC increment.
FLUSH	Scan path (Turns on A & B Clocks simultaneously).
SIGNDIVT	Sign divide macro logic and left shift.
ENHANCE1	MC-THRU AXUX, breakpoint logic, and MC Increment.
SMPY2	Sign multiple macro logic and right shift.

The results after fault simulation are detailed in Table 3.9.8-3. The first two columns show a comparison between good machine simulation and fault machine simulation. As the test percentage increases, the ratio of fault machine simulation time to good machine simulation time exhibits a downward trend indicating that fewer fault machines must be simulated.

Columns three and four of Table 3.9-4 show the percent detected and possibly detected by the fault simulator after each pattern set. These percentages are the cumulative test coverage since the results from previous sets were saved. Most of the possibly tested faults are due to uninitialized latch values. Both columns appear to be asymptotically approaching a percentage in the low to mid 80's. Conventional wisdom suggests that a good functional pattern set will test approximately 85% of the logic. If the fault simulation had included the entire functional test, the percentage tested would more than likely be close to that value.

ALU1 & ALU2, FLUSH, and SIGNDIVT were all fault simulated in the staticsin mode. This means that all patterns were converted to patterns with a single input change per pattern. This greatly increased the number of patterns which were generated and which had to be fault simulated. Since the functional tests do not exercise the logic in an LSSD test mode (i.e., one clock on at a time interspersed with scan ins and scan outs), allowing multiple input changes per pattern can cause the actual results to be different from the simulated results. Therefore, any manually generated test patterns which are to be released to manufacturing within IBM are required to be single input change per pattern.

As a result of this requirement, the CPU time and number of patterns shown in Table 3.9.8-3 are significantly understated since four of these pattern sets were not simulated in the staticsin mode. These pattern sets represent less than one sixth of the entire set of BDL/C functional pattern sets (Table 3.4-1). By simple linear extrapolation, completion of the fault simulation for the entire functional set would involve more than three hours of CPU time and require more than 7000 patterns to achieve a

Table 3.9.8-3. Fault Simulation

BDL/C	168 CPU Minutes		% Detected	% Possibly Detected	Number of Patterns
	Gd. Mach. Simul.	Ft. Mach. Simul.			
ALU1 & ALU2*	0:23	14:30	41.9	46.1	514
TEST/ENHANCE2	0:22	5:15	54.6	49.4	95
FLUSH*	0:08	0:56	60.1	65.5	70
SIGNDIVT*	0:13	4:00	68.4	73.7	260
ENHANCE1	0:15	4:20	72.4	77.0	38
SMPY2	0:15	3:24	73.5	77.8	67
MANUAL TOTAL	1:36	32:25	73.5	77.8	1144
AUTO TEST GEN	-	56.03	97.5	97.5	2275

* This data reflects the fact that the manual fault simulator was in the serial input mode (STATICSIN) which increases CPU time and number of patterns generated. This mode is required for actual testers.

test percentage which is probably less than 90%. On top of that, if all the fault simulation was done as required, in the staticsin mode, these numbers would be even higher.

Obviously, the manual functional patterns, as they are written, are very inefficient compared to LSSD test patterns. With a little effort some manual pattern subsuming could be done to improve the testing efficiency. At present, the functional patterns sets are all written to begin from known initialized conditions. By carefully rewriting these patterns, they could be made sequentially dependent such that results from previous patterns could be used by the subsequent patterns instead of each pattern set starting from scratch.

Even with these improvements, additional time and effort would have to be spent writing patterns to test the additional 10% or 20% of the untested logic. Eventually the manual pattern efficiency could approach or exceed those generated automatically, but the amount of time and effort involved is hard to determine. Since both the LSSD automatic test patterns and the manual test patterns would have to be fault simulated in the end, the cost is common to them both. In almost all cases, the LSSD automatic test generators will be much faster and cheaper in generating test patterns for LSI/VLSI designs than any human attempting to do the same job.

3.10 TEST PATTERN GENERATION (NON-LSSD)

In order to provide a testability comparison between LSSD and non-LSSD design techniques, the scan string was deleted from the SCS481 design, and DFF's were substituted for the appropriate SRL's (Section 3.3). Next, test generation was attempted using the EDS non-LSSD test generators. In an attempt to normalize the results, an upper limit of fifty-six 370/168 CPU minutes was set. Two different versions of the non-LSSD SCS481 were tested. One version simply substituted standard DFF's for the SRL's; the second version used DFF's with set and reset lines tied to their respective primary input pin. A detailed comparison of the results from test generation is provided in the concluding subsections. The following sections explain the steps taken to achieve those results.

3.10.1 SET/RESET NON-LSSD TEST GENERATION

Test generation on the set/reset non-LSSD SCS481 utilized the methodology and test generators described in Section 3.9. Table 3.10-1 details the test generators used at each step of the effort. The abbreviations TS and PS stand for time slice and pattern slice respectively. MDEC is short for maximum decision number. FWD and REV refer to whether the test generator starts at the beginning of the fault test (FWD) or at the end (REV).

The first test generator applied to the design was RANDGEN. A test coverage of 38.8% was achieved in 10 CPU minutes. However, a warning message was issued stating that no clock flags were found indicating that RANDGEN would probably be ineffective. At present, the only way to pass clock flags to the test generators is by doing an LSSD rules check. Test generation experts elsewhere in IBM stated that they never use RANDGEN for non-LSSD parts since it tends to be ineffective. That is why no non-LSSD method for passing clock flags exists.

Since the non-LSSD design would have numerous LSSD violations, the rules check program time limit was set very low to avoid a large printout. The rules check program finished successfully with a warning that no A or B

Table 3.10-1. Set/Reset Non-LSSD Test Generation Steps

Test Generator	Options	Completion Reason	Percent Detected	168 CPU Min.	Saved	No. of Patterns	Comments
1. RANDGEN*	TS	TS	38.8%	10	NO	-	Warning message because clocks not flagged.
2. RANDGEN	TS	TS	45.5%	11:43	YES	261	With clock flagged.
3. FAULTGEN	TS, FWD MDEC=25	TS	63.9%	28:48	YES	1163	2 outputs untested.
4. FAULTGEN	TS, REV MDEC=25	TS	64.8%	3:52	YES	70	1 output untested.
5. SOFTG	NoTS, PS FWD	PS	84.8%	3:36	YES	380	1 output untested.
6. SOFTG	NoTS, PS FWD	PS	86.7%	3:20	YES	300	1 output untested.
7. FAULTGEN	No, FWD MDEC=50	Time=5min	86.7%	5:00	NO	194	Only 2 faults detected.
8. FAULTGEN	NoTS;REV	Time=5min	86.9%	5:00	NO	181	13 faults detected including all outputs.
9. PODEM	NoTS, PS	Error	-	-	NO	-	Did not run, no error code printed.
10. SOFT G	NoTS, NoPS REV	Time=5min	89.8%	5:00	YES	620	1 output tested.
SAVED TOTAL	-	-	89.8%	56:19	-	2794	90.6% Possibly tested.
TOTAL *	-	-	89.8%	66:19	-	3169	

* RANDGEN #1 is not included in the total figures due to clocks not being flagged.

Clocks or SRL's were found. Since only a system clock (C Clock) and DFF's were in the design, this warning was reasonable.

RANDGEN try number two shows the improvement that results from passing the clock flag. RANDGEN test coverage increased to 45.5% in just under twelve minutes. The test generation experts thought this was an extraordinarily high test coverage for RANDGEN on a non-LSSD part. The high coverage for RANDGEN probably reflects the fact that most of the logic is only one or two latches away from an output. Also, the set and reset line seem to increase the coverage.

FAULTGEN with the default options of TS, FWD, and MDEC=25 was selected as the next test generator. This decision was made because FAULTGEN should be able to quickly pick up the easy combinational faults and some of the sequential faults. After just less than a half hour, FAULTGEN increased the test coverage to 63.9% using 1163 patterns. Two of the primary outputs still had not been tested to both states.

FAULTGEN was rerun starting from the bottom of the fault list since the previous run had not reached the end of the list. This run picked up an additional 1.9% including one of the output states. This required just less than four CPU minutes and 70 patterns.

Since relatively little progress was made in the previous FAULTGEN run, SOFTG with the options NTS, PS, and FWD was selected as the next test generator. After three and one half minutes, SOFTG detected an additional 20% of the faults using 380 patterns. Again, one output was still untested.

Due to the "learning" feature of SOFTG, the EDS manuals recommend re-running SOFTG with the same options to pick up additional faults masked by the "learning" feature. Test try number six is this SOFTG retry. This run detected only 1.9% more faults in three minutes and twenty seconds and 300 patterns. The total saved CPU time at this point equaled 51:19. Since an upper bound of 56 minutes was set, all subsequent jobs were time limited to 5 minutes.

The relatively modest gains in the SOFTG retry necessitated making alterations to the default options for the test generators. In test try seven, the MDEC level for FAULTGEN was increased from 25 to 50. After five minutes, only two faults were detected. This ineffectual pattern set was not saved.

Next FAULTGEN with no time slice was tried. After five minutes, only thirteen faults were tested. Again, this pattern set was not deemed efficient enough to save.

Test try number nine used PODEM. This run ended abnormally, but no error message was printed. PODEM probably ended because no shift clocks or SRL's were in the design even though the design was designated as LSSD to pass the clock flags for RANDGEN. PODEM should work on a non-LSSD design if properly set up, but would probably be inefficient.

The final test generation attempt was made using SOFTG with options no TS, no PS, and REV. This combination was able to increase test coverage to 89.8% within the allotted five minute period creating 620 patterns. This result was deemed acceptable and was saved.

The total of all steps saved was an 89.8% test coverage with 2794 patterns using 56:19 CPU minutes. Only steps 7 and 8 were not saved for a total of 10 minutes unsaved CPU time. An additional 0.8% was possibly tested. Some of these possibly test faults are 3A s-a-0 in the DFF's (Section 2.5). The rest are probably due to uninitialized latch values. The primary output XWRRT s-a-1 was never tested by these patterns.

3.10.2 STANDARD NON-LSSD TEST GENERATION

The intention for test generation of the standard non-LSSD SCS481 without sets and resets was to use the same sequence of test generators as were used for the set/reset SCS481. Table 3.10-2 shows the actual test generation sequence used.

Table 3.10-2. Standard Non-LSSD Test Generation Steps

Test Generator	Options	Completion Reason	Percent Detected	168 CPU Min.	Saved	No. of Patterns	Comments
1. RANDGEN	TS	TS	27.6%	10:46	YES	261	with clock flagged.
2. FAULTGEN	TS, FWD, PS	PS	47.8%	15:36	YES	324	
3. FAULTGEN	TS, REV, PS	TS	47.8%	1:52	NO	-	No faults detected.
4. SOFTG	NoTS, PS FWD	PS	47.8%	2:00	NO	-	No faults detected.
5. FAULTGEN	NoTS, PS REV	PS	47.8%	2:00	NO	-	No faults detected.
6. FAULTGEN	NoTS, NoPS, FWD	Time=30min	51.6%	30:00	NO	832	Only 3.8% increase.
7. SOFTG	NoTS, NoPS, FWD	Time=15min	92.3%	15:00	YES	1283	44.8% increase all outputs tested.
8. SOFTG	NoTS, PS FWD	PS	93.5%	2:56	YES	620	
9. SOFTG	NoTS, NoPS, REV	Time=12min	95.9%	12:00	YES	2977	
SAVED TOTAL	-	-	95.9%	56:18	-	5365	96.7% possibly tested.
TOTAL	-	-	95.9%	92:10	-	6197	

RANDGEN is run first again. The test coverage of 27.6% is less than half the set/reset version. Almost eleven minutes were used to generate the 261 patterns for this percentage. This low coverage explains why RANDGEN is rarely used for non-LSSD designs.

FAULTGEN with the default options was once more the second test generator tried. FAULTGEN was able to increase the test coverage to 47.8% in about fifteen and a half minutes requiring 324 patterns. An important thing to notice is that FAULTGEN ended because it was unable to generate a new pattern within a specified period of time (PS). This contrasts to the first FAULTGEN run for the set/reset SCS481 which ended because no faults had been detected (TS). FAULTGEN apparently was having difficulty generating patterns for the standard non-LSSD SCS481 indicating that regular DFF's complicate pattern generation.

Just as in the previous example, FAULTGEN with the reverse option and the default SOFTG test generators were both tried next with the result that no faults were detected by either run. Re-running SOFTG with the same options would provide no additional advantage, so it was not attempted. Test generation appeared to have stalled at 47.8% whereas the set/reset SCS481 was at 86.7% after the same sequence of test generators. More drastic measures were deemed necessary.

FAULTGEN with no TS was tried with no success, so FAULTGEN with no TS and no PS was applied to the design. After the allotted thirty minutes, only 3.8% more faults were detected. This small increase was considered too small in relation to the CPU time consumed, so it was not saved.

Since FAULTGEN obviously offered no further advantage, SOFTG was set loose with no TS and no PS. An upper time limit of fifteen minutes was set to avoid burning up excessive CPU time. Astoundingly, the test coverage increased to 92.3% using 1283 patterns.

The previous try was so successful that SOFTG with the default options was used next. In just under three minutes, the coverage improved 1.2% with 620 more patterns. SOFTG with NoTS and NoPS was tested starting from the bottom of the fault list. An upper limit of 12 minutes was set in order not to exceed the 56 minute upper limit. This run resulted in the final coverage of 95.9% using 2877 more patterns.

The final results for all saved steps were a 95.9% coverage, 56:18 CPU minutes, and 5365 patterns. Steps three, four, five, and six were not saved for a total of 35:52 minutes of unsaved CPU time. Of the unsaved steps, only step six was able to generate any patterns at all. An additional one percent of the faults were possibly detected, again including fault 3A s-a-0 from the DFF's. All primary output faults were tested.

3.10.3 NON-LSSD TEST GENERATION COMPARISON

Both non-LSSD test generation attempts achieved reasonably good test coverage in the time allotted. The standard non-LSSD design proved to be more difficult to test since the default options had to be overridden completely to achieve any results above 50%. Even with no limitations, FAULTGEN could only achieve 51.6%. The standard non-LSSD test generation also included three test generation runs which were unable to detect any faults.

Judging by the success SOFTG with NoTS and NoPS achieved on the standard design, the test coverage of both parts probably would have improved if this test generator and options had been used earlier. SOFTG tends to generate more patterns per fault than FAULTGEN which is why FAULTGEN is usually used first. However, once SOFTG gets started, it appears to do a very good job of detecting the faults.

With the benefit of a little learning the standard non-LSSD test coverage exceeded that for the set/reset version. However, the standard version encountered more dead ends, more wasted CPU time, more manual intervention, and more patterns.

Close examination of Table 2.5-6 tends to indicate another potential cause for the difference in test coverage. The untested faults in the stack, especially 30 s-a-0, point to the fact that many of the latches were never loaded with a value through the data input, but were instead loaded only by the set or reset line. In fact, the test generator seems to pick the set or reset line the majority of the time when it is tracing back for a sensitized path and encounters a latch. The set/reset line, in this case, allows the test generator to develop a pattern much more easily, however, with two potential drawbacks are encountered.

The first problem is that many faults require latches to be in opposite states. Since all the set and reset lines were tied together, putting latches in opposite states with the set or reset line is impossible. The test generator, not being intelligent, would attempt to use the set and reset lines first; and then, after discovering this to be impossible, it would have to retry using the data inputs. This is obviously the case for some of the faults left untested in the stack.

A second problem with using the set or reset line is that it leaves the data input faults of the latch untested. Developing a test which forces a value through a latch may be more difficult, but most of the rest of the latch gets tested at the same time. Fault simulation time is definitely increased by the consistent use of the set or reset line because each of the data input faults must be simulated for each pattern until they are tested. Whether the increase in fault simulation time offsets the decrease in test generation time is hard to say.

In any case, tying all the sets and resets together probably does not improve testability to any great degree. Some scheme of alternately connecting every other set and reset line together to two set and two reset lines would probably have been more successful. Clearly, some form of guidelines and design analysis are necessary to achieve the greatest testability benefit for this approach. Even so, this technique probably cannot guarantee testability in a sequentially complex design.

AD-A135 899

LSI (LARGE SCALE INTEGRATED) DESIGN FOR TESTABILITY
FINAL REPORT OF DESIG. (U) IBM FEDERAL SYSTEMS DIV
MANASSAS VA R D GROVES ET AL. NOV 83 AFMAL-TR-81-1068

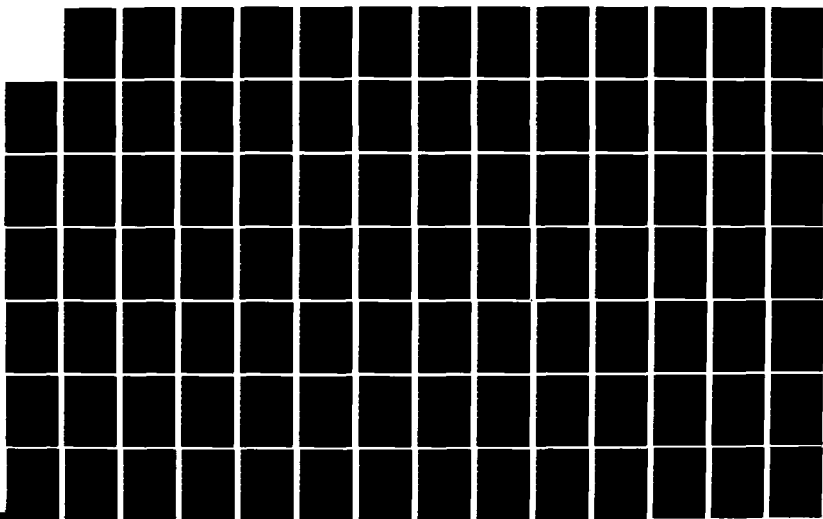
3/4.

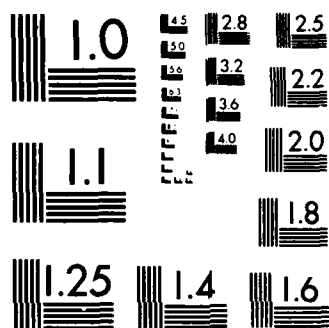
UNCLASSIFIED

F33615-79-C-1729

F/G 5/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Direct extrapolation of the standard non-LSSD design test coverage to that which would be achieved by the original T1 design should be made cautiously. As explained in Sections 3.1.3 and 3.3, several changes to the design were made which might have improved the testability. These changes involve the use of a more testable DFF model, the use of the Q output of that model, and the break apart and other changes to the B-Bus, OP8, and OP9 circuitry.

In achieving the 89.8% coverage for the set/reset version, 2794 patterns were necessary. To reach 92.3% (step 7) for the standard version, 1868 patterns were needed, however, an additional 3497 patterns were needed to increase this coverage to 95.9%. This indicates the difficulty involved in detecting those last few faults left in the non-LSSD designs. Caution should be taken in comparing these pattern numbers to the LSSD pattern numbers since, in both cases, these numbers represent the quantity of patterns generated automatically, not the number of patterns applied by the IC tester.

The net test coverage, after adjusting for redundant and reduced faults, is 88.3% for the set/reset non-LSSD version and 96.8% for the standard version. The reduction in the set/reset test coverage is due to the large percentage of reduced faults which were detected.

3.10.4 LSSD VS NON-LSSD UNTESTED FAULT COMPARISON

The fault comparison in Table 3.10.4-1 shows a breakdown of the total faults in the LSSD and non-LSSD designs. The reduced, redundant and test circuit faults are removed to allow a comparison based on net testable faults. The clock and latch faults are subtracted to show that the combinational logic faults are common to all the designs. The untested faults are similarly adjusted to yield the net untested faults.

The test generator test percentage is the test coverage figure given by the test generator fault simulation. This number is calculated by taking

Table 3.10.4-1. FAULT COMPARISON

	LSSD #1	LSSD #2	Non-LSSD	Non-LSSD w/sets & Resets
TOTAL FAULTS	6723	7046	7080	7394
Reduced Faults	(2351)	(2413)	(2719)	(2842)
Net Redundant Faults	(109)	(109)	(109)	(109)
Net Test Circuits Faults	-	(261)	-	-
NET TESTABLE FAULTS	4263	4263	4180	4442
Net Clock/Set/Reset	(138)	(138)	(65)	(121)
Net Latch Faults	(1000)	(1000)	(990)	(1196)
COMBINATIONAL FAULTS	3125	3125	3125	3125
Untested Faults	133	171	288	748
Reduced & Redundant	(127)	(135)	(154)	(227)
NET UNTEST FAULTS	6	36	134	521
Test Gen Test %	98%	97.4% ⁺	95.9%	89.8%
*Actual Test %	99.86%	99.16%	96.8%	88.3%
THEROETICAL TEST %	100%	100%	98.7%	98.8%
w/Q instead of \bar{Q}	-	-	98.2%	98.3%

* Actual Test % =

$$\frac{(\text{Total Faults} - \text{Reduced Faults} - \text{Redundant Faults} - \text{Net Untested Faults})}{(\text{Total Faults} - \text{Reduced Faults} - \text{Redundant Faults})}$$

+ LSSD #2 Test % = $\frac{(\text{Total Faults} - \text{Test Circuit Faults} - \text{Untested Faults})}{(\text{Total Faults} - \text{Test Circuit Faults})}$

@ Theoretical test percentage assumes that the only untestable faults are those untestable latch faults discussed in Section 2.5.1. This equals 54 faults. The second percentage assumes that the Q output of the DFF had been used instead of the \bar{Q} output. This increases the number of untestables to 76.

total faults minus untested faults divided by total faults (for LSSD #2 the test circuit faults are subtracted from the total faults to allow an equivalent comparison to the other three design which do not have test circuits). The actual test percentage is calculated by taking net testable faults minus net untested faults divided by net testable faults. All the actual test percentages are higher than the test generator percentage except for the set/reset non-LSSD. This percentage decreased because of the number of reduced faults detected; most of which do not correspond to physical faults.

The theoretical test percentage assumes all redundant faults have been removed and that the only untestable faults would be in the latches. Since the PH SRL is completely testable, the theoretical test percentage for the LSSD versions is 100%. The non-LSSD versions have 3C s-a-0 for all 22 non-stack latches; and 3C and 3A s-a-0 for all 16 stack latches producing 54 untestable faults. Therefore, the theoretically best test percentage for the non-LSSD designs is 98.7% and 98.8% respectively.

The DFF model used in the non-LSSD designs has an inverted \bar{Q} as its data output (see Figure 3.3-1). If the data output of the model had come from the Q output instead of the \bar{Q} , 3A s-a-0 would have been untestable for all 38 latches. This fact results in the second theoretical test percentage of 98.2% and 98.3%. Since normal designers would probably use the Q output instead of and inverted \bar{Q} , this percentage is probably more accurate. In either case, note that the actual test percentages for the LSSD designs are greater than the theoretical test percentages for the non-LSSD designs! This is due to the fact that the PH SRL is 100% testable.

Table 3.10.4-2 shows a breakdown of the net faults by location in the logic as well as separating combinational logic from clock and latch logic. Table 3.10.4-3 shows the same breakdown for the net untested faults. Table 3.10.4-4 contains the results obtained by dividing Table 3.10.4-3 by Table 3.10.4-2. Close examination of Table 3.10.4-4 reveals that all four test generation attempts achieved greater than 93% test coverage for the ALU, Data Registers, Instruction Decode, and Non-Stack Enhancements.

Table 3.10.4-2. NET FAULT BREAKDOWN

LOCATION	ALL FAULTS				COMBINATIONAL FAULTS				CLOCK/LATCH FAULTS			
	LSSD 1	LSSD 2	NON LSSD	S/R NON LSSD	LSSD 1	LSSD 2	NON LSSD	S/R NON LSSD	LSSD 1	LSSD 2	NON LSSD	S/R NON LSSD
ALU	1285	1285	1165	1184	1080	1080	1080	1080	205	205	85	104
ADDRESS REGISTERS	575	575	577	622	403	403	403	403	172	172	174	219
DATA REGISTERS	624	624	626	671	452	452	452	452	172	172	174	219
MACRO LOGIC	236	236	238	248	196	196	196	196	40	40	42	52
INSTRUCTION DECODE	804	804	804	804	804	804	804	804	0	0	0	0
NON-STACK ENHANCEMENT	279	279	274	302	181	181	181	181	98	98	93	121
STACK	460	460	496	611	9	9	9	9	451	451	487	602
TOTAL	4263	4263	4180	4442	3125	3125	3125	3125	1138	1138	1055	1317

Table 3.10.4-3. NET UNDETECTED FAULT ANALYSIS

LOCATION	ALL FAULTS				COMBINATIONAL FAULTS				CLOCK/LATCH FAULTS			
	LSSD 1	LSSD 2	NON LSSD	S/R NON LSSD	LSSD 1	LSSD 2	NON LSSD	S/R NON LSSD	LSSD 1	LSSD 2	NON LSSD	S/R NON LSSD
ALU	2	7	11	59	0	7	6	51	2	0	5	8
ADDRESS REGISTERS	0	10	13	97	0	10	2	89	0	0	11	8
DATA REGISTERS	0	6	28	41	0	6	16	26	0	0	12	15
MACRO LOGIC	0	8	5	47	0	8	3	42	0	0	2	5
INSTRUCTION DECODE	0	0	9	56	0	0	9	56	0	0	0	0
NON-STACK ENHANCEMENT	1	2	13	15	0	1	2	11	1	1	11	4
STACK	3	3	55	206	0	0	3	4	3	3	52	202
TOTAL	6	36	134	521	0	32	41	279	6	4	93	242

TABLE 3.10.4-4 FAULT COVERAGE BY LOCATION

LOCATION	ALL FAULTS					COMBINATIONAL FAULTS					CLOCK/LATCH FAULTS				
	LSSD 1	LSSD 2	NON	S/R	NON	LSSD 1	LSSD2	NON	S/R	NON	LSSD 1	LSSD 2	NON	S/R	NON
	LSSD	LSSD	LSSD	LSSD	LSSD	LSSD		LSSD	LSSD	LSSD	LSSD		LSSD	LSSD	LSSD
ALU	99.8	99.5	99.1	95%		100	99.4	99.4	95.3		99%	100	94.1	92.3	
ADDRESS REGISTERS	100	98.3	97.7	84.4		100	97.5	99.5	77.9		100	100	93.7	96.3	
DATA REGISTERS	100	99%	95.5	93.9		100	98.7	96.5	94.2		100	100	93.1	93.2	
MACRO LOGIC	100	96.6	97.9	81%		100	95.9	98.5	78.6		100	100	95.2	90.4	
INSTRUCTION DECODE	100	100	98.9	93%		100	100	98.9	93%		-	-	-	-	
NON-STACK															
ENHANCEMENT	99.8	99.3	95.3	95%		100	99.4	98.9	93.9		99%	99%	88.2	96.7	
STACK	99.3	99.3	88.9	66.3		100	100	66.7	55.6		99.3	99.3	89.3	66.4	
TOTAL	99.9	99.2	96.8	88.3		100%	99%	98.7	91.1		99.5	99.6	91.2	81.6	

The most significant difference between the versions occurs in the highly latch intensive stack. In this area LSSD achieved 99.3% tested and the non-LSSD versions were 88.9% and 66.3% respectively. This is the only location in the logic where the sequential depth is greater than two latches. LSSD is obviously superior in this sequentially complex circuit, where scanability can be used most advantageously.

The testability of the PH SRL becomes an obvious advantage when the clock and latch fault percentages are examined. The LSSD designs tested between 99% and 100% of these faults. In fact, all the untested faults were clock faults; 100% of all latch faults were tested. Due to the inherent untestables in the DFF's as well as the difficulty involved in detecting their sequentially dependent faults, the test percentage in the non-LSSD designs never exceed 96.7% and averaged less than 92%. If the CSR SRL or a "glitchless" PH SRL had been used as the LSSD SRL, the latch test percentage would have been less than 100%; but overall, probably would still have been higher than the non-LSSD version, especially in the stack.

The non-LSSD version without sets and resets actually achieved a higher test percentage for the combinational logic in the Address Registers and the macro logic. Table 3.10.4-5 shows that these two sections of logic have the highest percentage of redundant logic. In fact, this table shows a definite trend for decreasing test percentage with increasing redundancy percentage. This would seem to indicate that the test generators do not do as well because they must spend time trying to generate tests for faults which are not testable. This could be the reason that the non-LSSD test coverage was better; however, the difference is so small that it might be attributed to chance.

3.10.5 LSSD VS NON-LSSD TEST GENERATION CONCLUSIONS

The lack of sequential depth in the standard 74S481 design simplifies automatic test generation since over 70% of the faults are combinational in nature. This fact is corroborated by the similar test coverage percentages obtained in both the LSSD and non-LSSD combinational faults.

TABLE 3.10.4-5 COMBINATIONAL TEST % TO REDUNDANCY COMPARISON

LOCATION	REDUNDANT FAULTS	COMBINATIONAL FAULTS	TOTAL FAULTS	PERCENT REDUNDANT	AVERAGE PERCENT TESTED
ALU	34	1080	1114	3.05%	98.53%
INSTRUCTION DECODE	26	804	830	3.13%	97.98%
DATA REGISTERS	17	452	469	3.62%	97.43%
ADDRESS REGISTERS	20	403	423	4.73%	93.73%
MACRO LOGIC	12	196	208	5.77%	93.25%
NON-STACK ENHANCEMENT	0	181	181	0%	98.1%
STACK	0	9	9	0%	80.58%
TOTAL	109	3125	3234	3.37%	97.2%

However, as pointed out in Section 3.1.3 and 3.3, direct comparison of the testability of the SCS481 non-LSSD designs to the 74S481 non-LSSD design is not completely applicable since potential enhancements to the testability of the SCS481 versions were included.

Even though the non-LSSD test coverage figures compare favorably with the LSSD coverages, the non-LSSD attempts are prone to chance and involve much learning, intervention, and computer time. This additional effort differentiates LSSD and non-LSSD test generation. This difference tends to increase exponentially with increasing sequential complexity. Comparing the non-LSSD efforts with LSSD pass #1 provides a sharp contrast between the two approaches. In less than 32 minutes, basically 100% of the faults were detected. By linear extrapolation, both non-LSSD test generation attempts tested less than 65% of the logic after 32 minutes.

LSSD's advantages become obvious in the latch fault coverage, especially in the stack logic. These advantages accrue from both the greater testability of the level sensitive PH SRL and the scanability of these latches. Coupled with the benefits of totally automatic test pattern generation and subsumption, the fact that an in depth design testability analysis is unnecessary - provided the design rules are followed - and that the technique is universally applicable to all designs from the simplest to the most complex, LSSD provides significant advantages for design testability, especially as sequential complexity and gate to pin ratios increase.

3.11 RELEASE DATA TO MANUFACTURING

After the physical packaging and testing have been completed, the EDS user releases the design, its documentation, and the test patterns that will be used to test the actual hardware--all in a digital form written on a RIT (Release Interface Tape).

The release process involves selecting design data from a CID and/or from other previously generated data sets, and writing the RIT for manufacturing. The RIT includes, in a series of contiguous logical files: engineering change information; logic design data; package descriptions; physical design records; test data, and ALD images. After the RIT is given to manufacturing, the recorded data is reformatted by a manufacturing system for driving automated manufacturing equipment.

In order to prepare the design for writing the RIT in EDS, users add bill of material and part number data to the design during the release phase of the design cycle. The number of files and the type of data on the RIT depend on the product technology and the package level being processed.

The design data that is released is mostly physical design data. The only logical design data that is of value to the manufacturing process is the ALD representation. Most of the design data that is used by manufacturing is created during physical design so the release process is essentially an extension of the physical design process.

3.11.1 PRE-RELEASE CHECKING (AUDIT)

As with each LSI design, each LSI manufacturing process requires verification data as a part of the release package. This data assures manufacturing that the design can be produced successfully. One such verification device is a set of flags called completion codes.

A completion code is set by each physical design action to indicate how successfully that action was completed. These codes work as check-

points during the physical design process, and are checked during the first phase of release activity after all physical design has been completed. If any completion code indicates that a physical design action was not successful, the design process may have to be restarted at that point.

Completion codes are "fed back" to the CID and become part of the design data that is released to manufacturing. The release processing system verifies these codes and other data to ensure that all required physical design steps were completed.

3.11.2 INPUTS TO THE RELEASE PROCESS

The RIT contains a variable number of files of design data, depending on the type of unit (card, board, module, or chip) being released. This data may be one or more of the following:

- (a) Updated CID (ALDs, physical design data, part and engineering change numbers).
- (b) TDF (Test Data File).
- (c) Comments data.
- (d) Release control statements.
- (e) Physical design rules.

Another way of describing this input data is as control information, build data, test data, and design documentation. Control information which must be generated includes bills of material, part numbers, and EC levels.

Build data describes where the components are to be placed and the routing of embedded and printed wires. Test data includes the data patterns accumulated during test data generation. Product documentation

includes the ALDs, messages, and comments the designer wishes to forward to manufacturing. Each of these four types of information are produced by EDS design applications.

3.11.3 PREPARING TEST DATA FOR RELEASE

Once the design has had tests generated satisfactorily, the associated test input and output patterns are recorded on the fault simulation output file, or AET file (Section 3.5.3). The AET is functionally the same output file used to record the results of logic design verification. In this case, however, the desired output is just the test data. This data is extracted from the AET and is compiled along with failure data and other related information, into the Test Data file (TDF) for release to manufacturing.

3.11.4 PREPARING PHYSICAL DESIGN DATA FOR RELEASE

Physical design data such as placement and wiring is simply "fed back" onto the CID upon completion of Physical Design. The RIT generation program then extracts from the CID the physical design information which must be included in the RIT. When all the information required for the RIT has been generated and placed on the RIT, the tape can then be sent to the manufacturing site for fabrication.

3.12 FABRICATION AND TEST

Fabrication of the SCS481 was completed at IBM's facility at East Fishkill. The fabrication begins with the reception of the RIT. The RIT is then audited to assure that all the necessary information is included on the RIT and that the information is valid and up to date. The SCS481 RIT was accepted by East Fishkill on February 25, 1980 and good samples were available May 16, 1980.

Masks are then generated from the physical design data on the RIT. Once the masks have been created, the metal personalization of the wafers can begin. When the wafers have completed all personalization and processing steps, each chip on the wafer is tested using the patterns from the TDF on the RIT plus some technology dependent parameter tests added by East Fishkill.

The wafers are then diced and the good chips are joined to the module substrate. Modules can be either dry capped or encapsulated. Dry caps merely have the aluminum top crimped onto the module. The encapsulated modules are potted and baked in an oven to seal them. This takes two or three additional days, so the initial good SCS481's were dry capped. Subsequent SCS481 chips were encapsulated.

3.13 TEST DIAGNOSTIC SELECTION

Failed devices were selected by a procedure that involved both Manassas and East Fishkill personnel. The good devices were automatically picked, packaged and delivered. The failed devices consisted of those that failed the initial screening and those that failed functional (LSSD test pattern) tests. Devices which fail screening are usually so grossly bad that they are discarded immediately. The devices which made it through screening, but failed functional test were retained by test engineering.

3.13.1 FAILED DEVICE SELECTION

When all of the chips were in final test, a trip to East Fishkill was taken to select the partial good devices to be used for the demonstration. Device selection was made primarily on the basis of failed pattern number. Figure 3.13-1 is a sample printout of the failed device data taken in East Fishkill. Each chip is identified by wafer number and X Y coordinates. Included is the pattern number failed and the faulty measured levels and their respective pins.

The 28 chips which failed pattern 41 or higher were selected, as well as two chips which failed the flush test. Twenty-eight of these chips survived module joining. A list of these partial good chips and the pattern which they failed is provided in Table 2.2.2-1. Chips #2, #4, and #12 failed earlier patterns after chip join than they did on the wafer. This indicates that the chips were either not joined properly or another error was induced internally by the chip join process.

3.13.2 FAILED DEVICE ANALYSIS

Using the data provided by the tester in East Fishkill shown in Figure 3.13-1, an analysis program in East Fishkill examines the failed output pattern and the applied input pattern to determine the stuck faults which could cause the observed failure. The program eliminates those faults which should have been detected by previous patterns. A listing of possible faults is then printed for future analysis by the test engineer.

P/N	MKT	WF	X	Y	P/F	TYPE	PTN	PA	MSRMT	HIGH	LOW	PA	MSRMT	HIGH	LCN	PA	MSRMT	HIGH	LOW
2844324	11	12	4	8	F	NUM	127	28	0.446V			4A	2.225V						
	39	8	12	F	NUM	127	28	0.445V				4A	2.211V						
	73	6	10	F	NUM	127	28	0.436V				4A	2.212V						
	68	10	8	F	NUM	131	19	0.434V											
	03	6	4	F	NUM	132	7	2.213V				4A	0.438V						
	58	9	9	F	NUM	134	19	2.223V											
	32	4	7	F	NUM	135	28	2.243V				4A	0.412V						
	12	9	6	F	NUM	139	3A	0.440V											
	53	7	4	F	NUM	139	18	2.232V											
				F	NUM	139	59	2.222V				19	2.223V			34	0.426V		
	50	11	3	F	NUM	147	19	0.451V				5C	2.220V						
	41	8	2	F	NUM	155	18	2.228V				34	2.120V			5C	0.440V		
	03	5	7	F	NUM	171	41	2.125V				22	2.227V			34	0.447V		
				F	NUM	171	5C	0.425V				42	2.121V			57	0.450V		
	39	13	8	F	NUM	171	18	0.433V											
	68	13	6	F	NUM	171	18	0.426V				42	2.121V			54	0.441V		
	39	5	4	F	NUM	183	18	0.436V				19	0.428V			54	0.445V		
				F	NUM	183	52	2.132V				55	2.116V			35	0.433V		
	96	6	4	F	NUM	183	52	2.135V											
	73	13	4	F	NUM	186	4A	0.443V			0.600V	4A	0.442V						
				F	NUM	186	4A	0.441V			0.200V	4A	0.441V			4A	0.442V		
				F	NUM	186	4A	0.442V				4A	0.441V			4A	0.441V		
				F	NUM	186	4A	0.442V				4A	0.440V			4A	0.441V		
				F	NUM	186	4A	0.441V				4A	0.441V			4A	0.441V		
	17	7	12	F	NUM	211	5C	0.432V											
	63	8	3	F	NUM	235	59	2.221V				19	0.435V			32	0.430V		
	41	5	9	F	NUM	247	18	0.436V				55	0.443V			5C	0.423V		
				F	NUM	247	34	2.121V											
				F	NUM	247	8	2.130V											
	50	3	10	F	NUM	279	23	0.424V				59	0.417V			35	2.241V		
	32	4	8	F	NUM	295	18	0.411V				19	2.244V						
				F	NUM	295	54	0.418V											
	03	9	12	F	NUM	335	19	0.420V				32	0.426V			34	2.121V		
				F	NUM	335	52	0.433V				54	0.440V			55	0.440V		
	17	6	5	F	NUM	335	52	0.449V				55	0.450V						
	32	9	12	F	NUM	335	19	0.403V				32	0.400V			34	2.121V		
				F	NUM	335	52	0.411V				54	0.420V			55	0.420V		
	41	8	3	F	NUM	335	55	0.455V											
	03	5	6	F	NUM	359	19	2.220V				21	2.225V			43	0.441V		
				F	NUM	359	5C	0.421V											
	50	7	6	F	NUM	367	18	0.450V				54	2.218V			59	0.439V		
	46	11	9	F	NUM	399	19	0.426V											
	12	13	4	F	NUM	427	3A	0.434V											
				F	NUM	491	18	0.427V				22	0.437V			23	2.222V		
				F	NUM	491	43	2.225V				54	2.224V			5C	2.225V		
	50	5	9	F	NUM	535	19	0.436V				21	0.450V			23	2.220V		
				F	NUM	535	32	0.440V				35	2.218V			43	2.218V		
				F	NUM	535	54	0.455V				59	2.220V			5C	2.226V		
				F	NUM	547	59	2.225V											
	03	13	11	F	NUM	610	19	2.221V				21	2.227V			34	2.122V		
	69	10	2	F	NUM	616	23	0.458V				32	0.441V						
	17	7	6	F	NUM	616	59	0.458V											
				F	NUM	628	13	0.432V				22	0.422V			32	0.425V		
	52	14	6	F	NUM	628	44	2.121V				54	0.444V						

A sample printout for partial good chip #14 is shown in Figure 3.13-2. Using the data from this program, the probable cause of failure for most of the partial good chips were determined. The results of this analysis are reflected in Table 2.2.2-1.

PATN PIN SCFF CICTICNARY LINE

676 512-26 1285 1444 4012 4165 4353 4355 4498 4958 6504 6574

FAULT #	EDSNAME	TYPE	LCC
1	EDSNAME	TYPE	LCC

1285	PR177C120	SA0	4A1-P03	BM2-P0C	BM4-P02	BM1-P01
4012	PR227BF60	SA0	BM1-P6C			
4165	PR227CE010	SA0	BC1	BC1-P02	BM4-P02	
4335	G000000006	SA0				
4335	PR110C860	SA0	BM1-P6C			
4950	PR110CE00	SA1	BM1-P6C			
4950	PR158B010	SA1	BM1-P6C			
6574						

O-X SIMULATION FOR PN2844325 WITH NPAT=676
SNIPIC ANALYSIS IS PERFORMED WITH CUTOFF=20

[illegible]

3.14 TEST CIRCUIT BOARDS

Test circuit boards were constructed for mounting and connecting the 74S481 and SCS481 to a test vehicle. The test vehicle was used to demonstrate the operation of the level sensitive SCS481 in an edge triggered environment.

3.14.1 TEST CIRCUIT BOARD FOR 74S481

A test circuit board was fabricated for the purpose of connecting the original 74S481 chip to the test vehicle. A socket was constructed for the 74S481 Quad-In-Line chip using 12-pin strips with 100 mil centers. This socket was mounted on a breadboard and a cable was designed to connect this board to the 74S481 socket on the test vehicle. A known good 74S481 was mounted in the socket on the breadboard. The intent of this test circuit board was to provide a means for connecting a 74S481 to the test vehicle without actually inserting the 74S481 into the socket. This was deemed necessary because the Quad-In-Line pins of the 74S481 are extremely fragile and susceptible to damage with frequent handling. The 74S481 is permanently installed in the 74S481 circuit board and the cable termination, of sturdier construction than the 74S481 pins, is used to connect the 74S481 to its socket on the test vehicle. An outline drawing of the 74S481 test circuit board is shown in Figure 3.14-1 and a schematic of the board is shown in Figure 3.14-2.

3.14.2 TEST CIRCUIT BOARD FOR SCS481

The SCS481, a 1.7 volt power supply regulator module, a 1.7 volt power supply transistor, five 54XX type integrated circuits for certain logic signal inversions, switches for slice position selection, and various resistors and capacitors are contained on the SCS481 test circuit board. This board was cabled to the 74S481 socket on the test vehicle using the same cable as the one used for the 74S481 test circuit board. This cabling technique was designed to permit connecting the 74S481 into the test vehicle for demonstration purposes or for assistance in isolating problems.

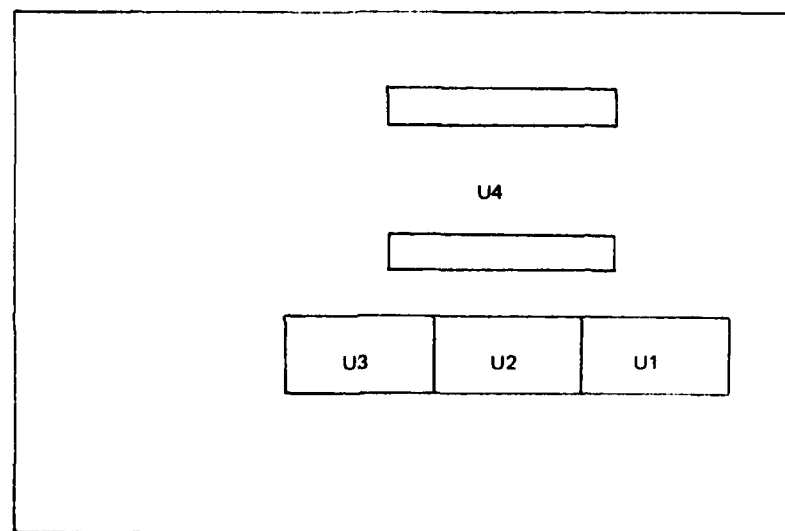


Figure 3.14-1. 74S481 Test Circuit Board Outline Drawing

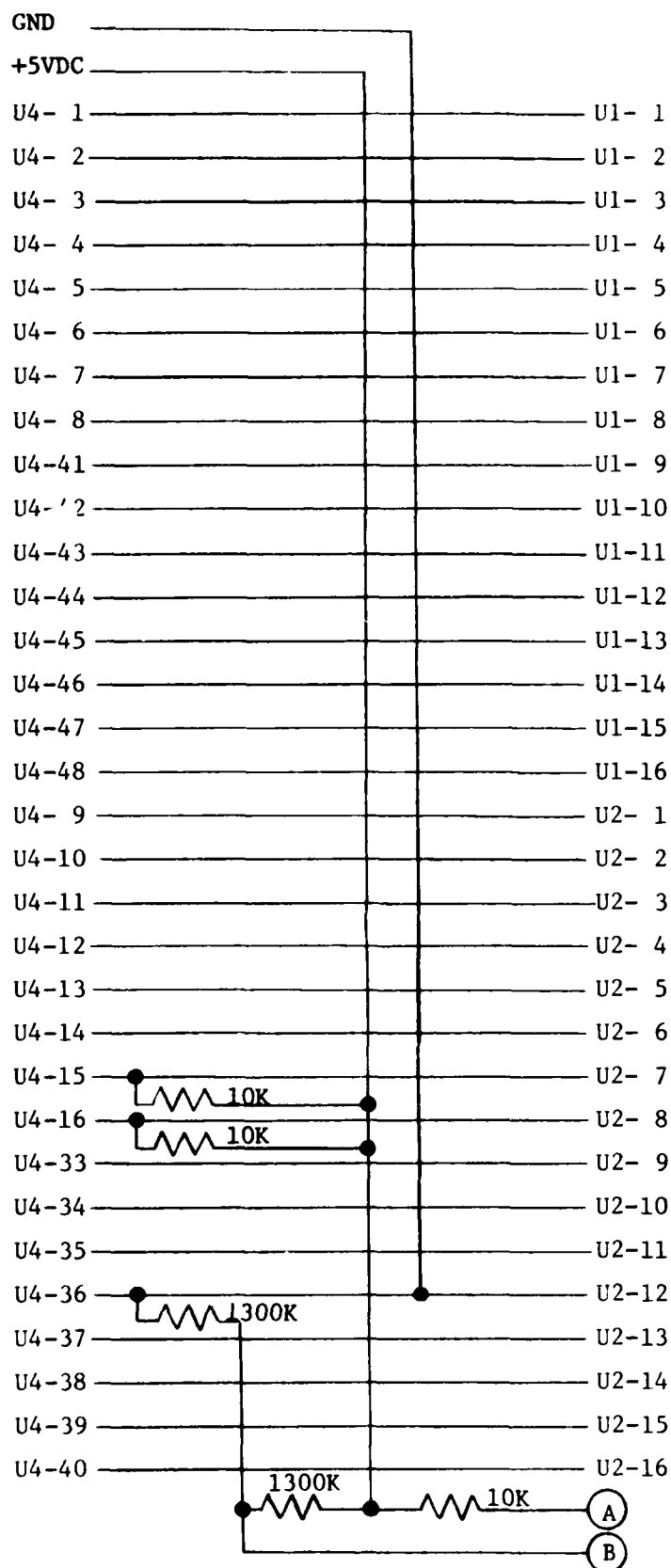


Figure 3.14-2. 74S481 Test Circuit Board Schematic

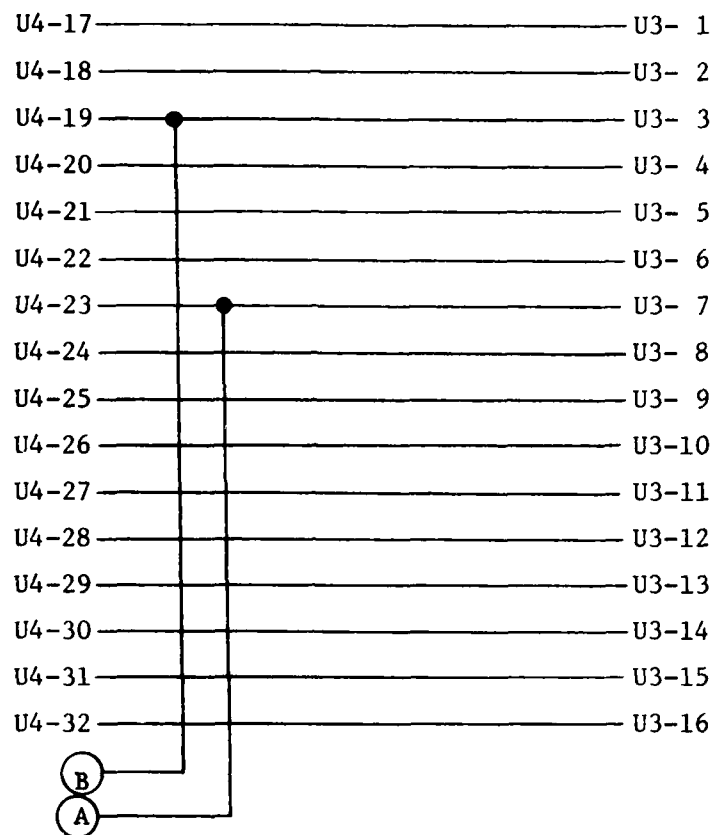


Figure 3.14-2. 74S481 Test Circuit Board Schematic (Continued)

The SCS481 was connected to the SCS481 test circuit board with a ZIF (zero insertion force) socket. The ZIF socket permitted easy insertion of good and bad modules in order to exercise the test vehicle functional tests on the SCS481's. An outline drawing of the SCS481 test circuit board is shown in Figure 3.14-3 and a schematic of the board is shown in Figure 3.14-4.

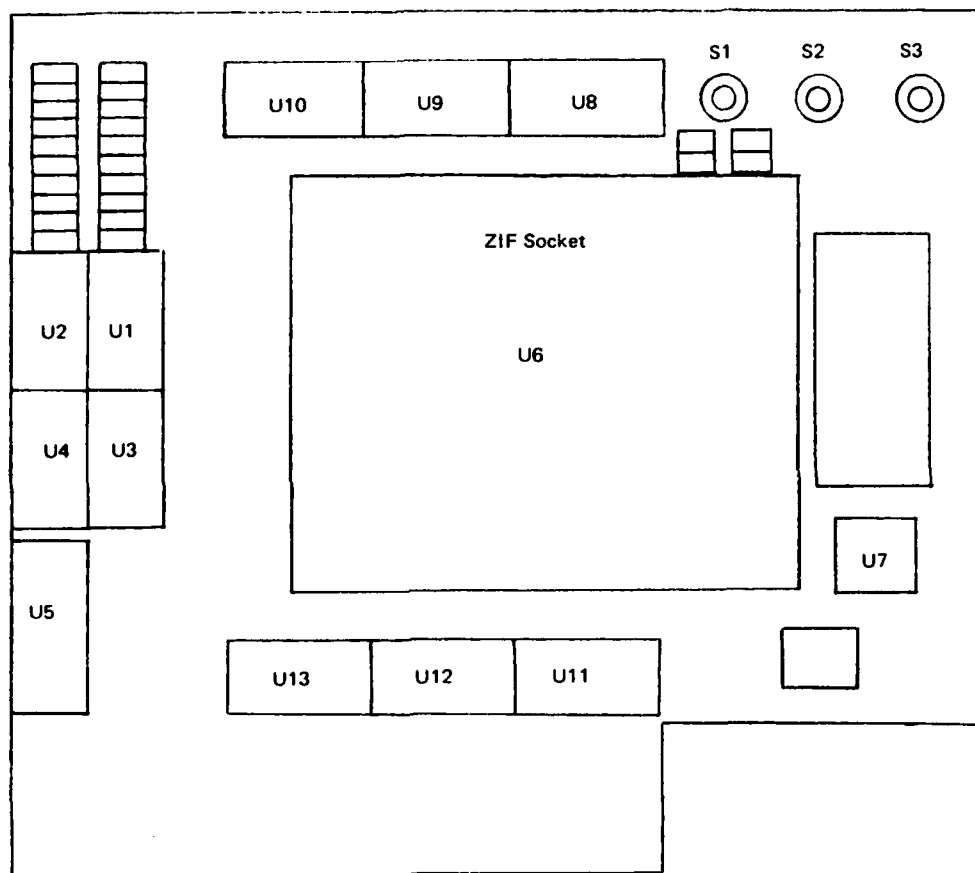


Figure 3.14-3. SCS481 Test Circuit Board Outline Drawing

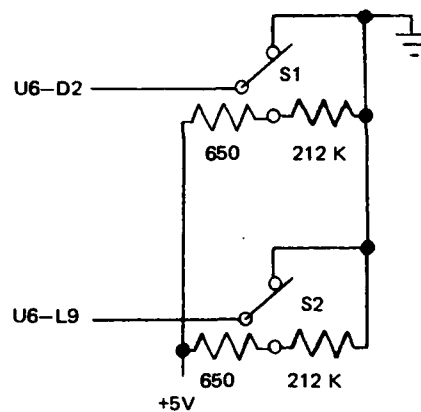
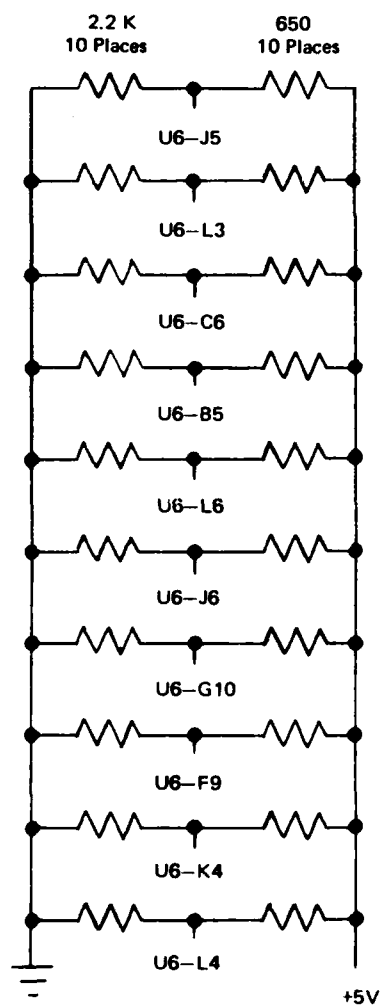
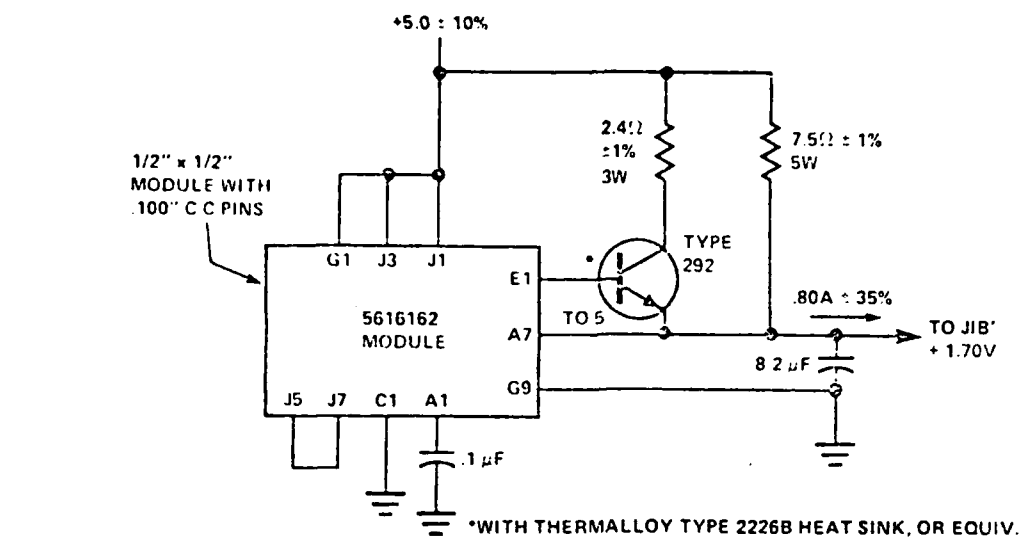


Figure 3.14-4. SCS481 Test Circuit Board Schematic

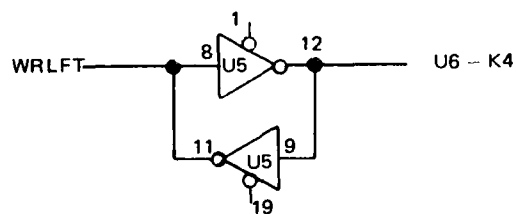
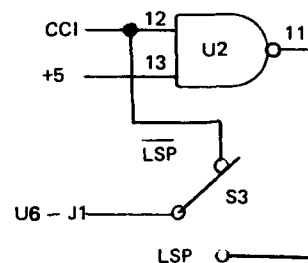
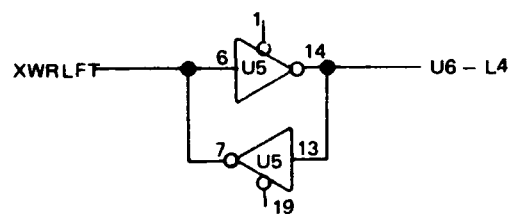
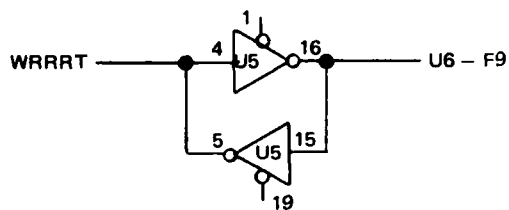
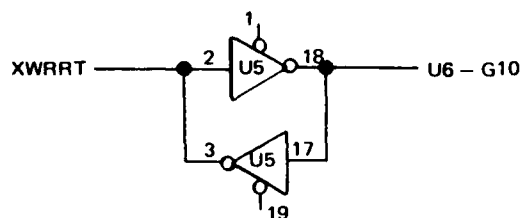
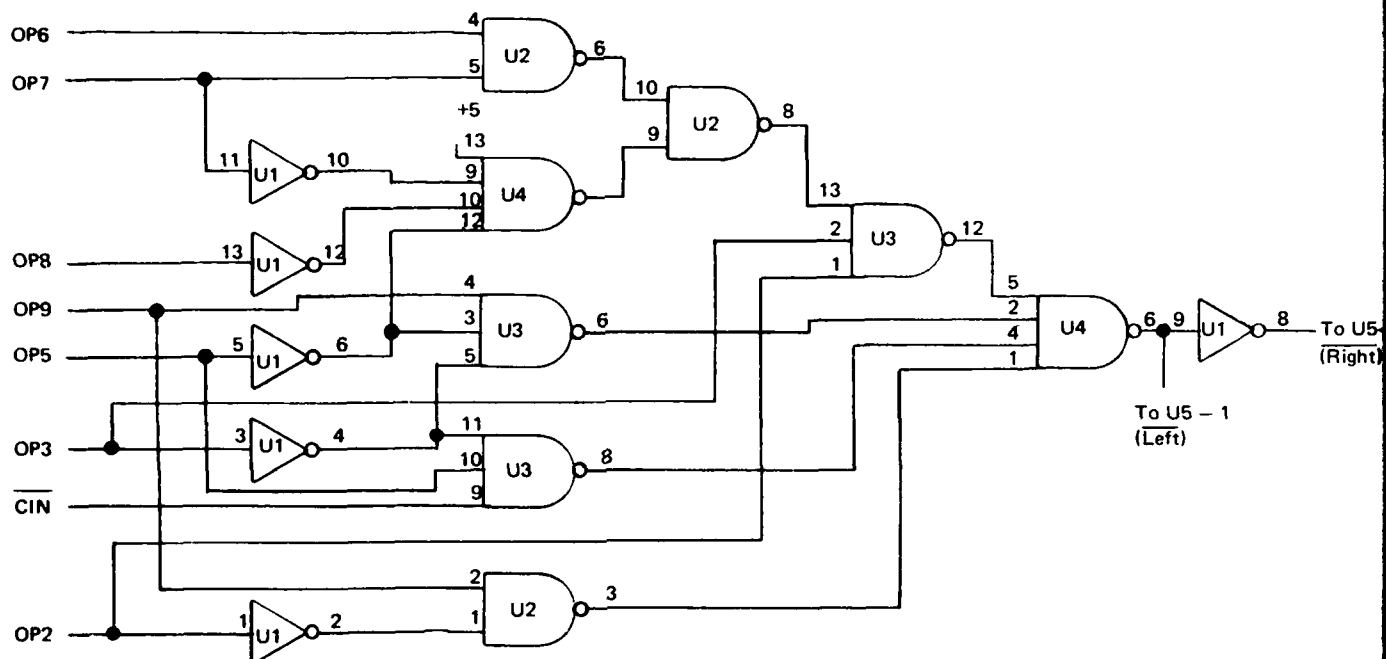


Figure 3.14-4. SCS481 Test Circuit Board Schematic (continued)

U10- 1	U6-B9
U10- 2	U6-J4
U10- 3	U6-F2
U10- 4	U6-F1
U10- 5	U6-F3
U10- 6	U6-J6
U10- 7	U6-L6
U10- 8	U6-B5
U10- 9	U6-C6
U10-10	U6-A9
U10-11	U6-B8
U10-12	U6-G8
U10-13	U6-C7
U10-14	U6-A8
U10-15	U6-A6
U10-16	U6-A7
U9 - 9	U6-C8
U8 - 1	U6-G9
U8 - 2	U6-J8
U8 - 3	U6-J9
U8 - 4	U6-E10
U8 - 5	U6-F8
U8 - 6	U6-K11
U8 - 7	U6-D10
	U6-D2*
	U6-L9*
U8 - 8	U6-B3
U8 - 9	U6-J10
U8 -10	U6-E11

* See first page of Figure 3.14-4.

Figure 3.14-4. SCS 481 Test Circuit Board Schematic (Continued)

U8-11	U6-H8
U8-12	U6-K9
U8-13	U6-K10
U8-14	U6-C5
U8-15	U6-K8
U8-16	U6-B2
U9-11	U6-D11
U9-12	U6-C11
U9-13	U6-E9
U9-14	U6-D5
U9-15	U6-C10
U9-16	U6-C9
U11-1	U6-B10
U11-2	U6-C4
U11-3	U6-C3
U11-4	U6-D1
U11-5	U6-D4
U11-6	U6-C2
U11-7	U6-H2
U11-8	U6-F4
U12-1	U6-G2
U12-2	U6-E3
U12-3	U6-K7
U12-4	+5V
U12-5	U6-E1
U12-6	U6-G1
U12-7	U6-L3
U12-8	U6-E4
U13-1	U6-D3
U13-2	U6-G3
U13-3	Unused

Figure 3.14-4. SCS 481 Test Circuit Board Schematic (Continued)

U13- 4	U6-K5
U13- 5	U6-L5
U13- 6	U6-H6
U13- 7	U6-J5
U13- 8	U6-B1
U13- 9	U6-K4
U13-10	U6-F9
U13-11	U6-L4
U13-12	U6-G10
U13-13	U6-H10
U13-14	U6-J11
U13-15	U6-G11
U13-16	U6-H9
U12- 9	U6-F10
U12-10	U6-F11
U12-11	U6-H1
U12-12	Gnd
U12-13	U6-E2
U12-14	U6-B6
U12-15	U6-A5
U12-16	U6-D6
U11- 9	U6-B7
U11-10	U6-A3
U11-11	U6-C1
U11-12	U6-J1*
U11-13	U6-H11, L8
U11-14	U6-B4
U11-15	U6-A4
U11-16	Unused

* See first page of Figure 3.14-4.

Figure 3.14-4. SCS 481 Test Circuit Board Schematic (Continued)

3.15 MDS-800 TEST CONFIGURATION

The Intel Microcomputer Development System, MDS-800, was programmed to perform preliminary evaluation of the SCS481 parts. This exercise was also intended to check out the test circuit board which contained the SCS481 zero insertion force socket and related support circuitry.

The MDS-800, which is an 8080 microprocessor based system, consists of a CRT/Keyboard, Floppy Disk Controller, Line Printer, and a main cabinet which has 18 card slots used to house a CPU card, memory card(s), and various cards used to control printer, CRT, floppy disk, etc. Each of the cards connects to a common 86 line signal bus which contains all necessary address, data, clock, voltage, and control signals which enable the various circuit cards to communicate with each other.

Most of the 18 cards slots in the MDS configuration as shipped by Intes are vacant. Some of these have been used to hold SBC116 and SBC508 I/O expansion boards. The I/O expansion boards provide the means by which the 8080 microprocessor can output data to an external device or read data from the device. The 8080 microprocessor places an address on its address lines to select a particular I/O expansion board and one of the I/O ports on that board. Then, if performing output, it puts a data item in its accumulator and performs a write operation. If the 8080 is performing a read operation, it addresses the I/O port and reads data into the accumulator.

The signal pins on the SCS481 were assigned to input or output port lines on the I/O expansion cards. Cables were fabricated to connect the SCS481 Test Circuit Board to the I/O expansion boards. Programs were written for the 8080 microprocessor to set up known values at the inputs to the SCS481 and read the response at the Data Out Port, the Address Out Port, and the other output pins. This program not only checks out the SCS481 chip but also exercises the other components on the SCS481 test circuit board with the intention of uncovering any opens, shorts, or incorrect wiring before taking the parts to Owego for functional testing.

The MDS-800 tests seven basic functions of the SCS481. First the working register is loaded and read back to verify the data written. Next a NOOP (OP Form IX) subtest is performed to test that EQ and Y are forced high, and X is forced low. Then an OP For IA subtest is performed that tests load/read A, load/read B, carry look ahead, carry out, equals, and add functions. The OP Form IB subset loads and reads the memory counter, working register, and the extended working register. Next an increment subtest checks the logical functions (OR, NOR, XOR) of the 74S/SCS481. Finally OP Forms IV, V, and VI are tested with a shift subtest. The seven MDS basic function subtests are broken into two basic tests for ease of testing and troubleshooting problems. The MDS test breakdown is shown in Table 3.15-1. Upon detection of an error, the MDS-800 prints EXPECTED/ACTUAL data seen by the tester.

The 74S481 mounted on its test circuit board was first exercised on the MDS-800 to verify the tests written on the MDS-800. The intent of testing was to prove that the test software runs on a 74S481 before trying to test a SCS481. At this point the I.O signal inversions of the SCS481 were discovered and corrected before any SCS481 hardware had been delivered. After software debug, the MDS-800 tests were successfully executed with the 74S481 connected to its test circuit board and the test circuit board connected to the MDS-800.

Table 3.15-1 MDS-800 Test Description
74S/SCS481 Diagnostics (MDS-800)

CPU TESTS

- Test 1
 - o Working Register Load/Read
 - o OP Form IX Test (NOOP)
 - o OP Form 1A Test (A, B, X, Y, COUT, EQ ADD)
 - o OP Form 1B Test (MC, PC, WR, XWR)
 - o Increment Test
- Test 2
 - o Op Form VIII Test (Logicals -- NOR, OR, XOR)
 - o OP Forms IV, V, VI Tests (Shift)

3.16 DESCRIPTION OF TEST VEHICLE

The test vehicle used for demonstrating the SCS481 chips, was the AP-101C processor used in the B-52 G/H Offensive Avionics System (OAS). The AP-101C is a parallel binary general purpose processor with 16 bit internal data paths. The Central Processing Unit, which uses the 54/74S481, contains facilities for addressing main storage, for fetching or storing information, for arithmetic and logical processing of data, for sequencing instructions in the desired order, and for initiating the communication between storage and external devices.

The arithmetic section of the CPU performs 16 and 32-bit fixed point operations, single precision floating point operations, and extended precision floating point operations. Fixed point numbers are represented in twos complement form. Single Precision Floating Point (SPFP) numbers (32-bits) have a minimum mantissa precision of 23-bits not including sign after normalization. An Extended Precision Floating Point (EPFP) capability is provided. Numbers represented in the EPFP format have a minimum mantissa precision of 31 bits, not including sign, after formalization. The range of a floating point number is determined by the exponent which in both cases gives a range from 10^{-39} to 10^{38} .

Much of the ALU and main storage addressing logic is located on the CPR4 page which contains the 74S481s. For the purposes of this demonstration a CPU4 page was modified with a socket wired into the board at the physical position corresponding to the least significant 74S481 bit slice position. This permitted easy cabling of the SCS481 test circuit board.

The test exercise was performed at the IBM facility in Owego, N.Y. where the AP-101C computer was designed and is presently manufactured and tested. Control of the test sequence was accomplished by means of a Programmable Interface Unit (PIU). The PIU is used to control testing of the AP-101C. The PIU is a portable, self-cooled unit with an integral Processor Control Panel (PCP) and self-contained power supplies. An

integrated cartridge tape drive is provided for Initial Binary Load (IBL) of PIU and AP-101C. The PIU provides the real-time interface lines to the interface of the AP-101C that is used to load, control and monitor the AP-101C during testing.

A comprehensive description and detailed instructions on usage of the PIU can be found in the PIU users manual (IBM Document Number 78-290-011). This manual describes the controls, displays, and functional operations of the PIU for a user to program load, execute, control, and monitor the AP-101C Avionic Processor of B-52 G/H OAS. Operation of the PIU control panel, display panel, integrated tape cartridge drive, function switch setting, sequences and modes of operation are defined in detail for the user.

The software executed for testing the AP-101C is called the Factory Test Program (FTP). The FTP is used to verify the operation of the AP-101C computer. Control is provided from the PIU. The machine operates in a stand-alone environment when being tested. Extensive listing comments are provided to aid in program understanding or problem solving. Acceptance test, burn-in test, and qualification test for the AP-101C are all performed using FTP.

FTP first runs through MINOPS subtest where very basic operations are performed (load/read registers) followed by Initialization. Then the four basic subtests are executed--CPU subtest, Memory subtest, I/O subtest, and Special subtest. The subtest of concern for the SCS481 is the CPU subtest. It is further broken down into seven categories: Basic, Branch, Arithmetic, Logical, Shift, Byte, and Special. Upon a test failure, the PIU operator can locate the test failure at the microcode level to determine the particular function being performed when the failure occurred. The AP-101C and its FTP do not exercise the 481 OP Forms III, VII, XII, or XIV.

The 74S481 and its test circuit board were connected to the AP-101C and the test software (FTP) executed to verify the integrity of the connector and cable installed in the Least Significant Position (LSP) of the 16 bit AP-101C. Connections were made and FTP executed successfully, demonstrating the operability of the connector and connecting cable.

SECTION 4

RECOMMENDATIONS FOR LSSD DESIGN RULES IN A MILITARY ENVIRONMENT

The LSSD design rules were listed in Table 3.1.2-3. For LSI/VLSI chip design in the MIL environment, no specific additions or deletions to these rules are recommended except in the area of embedded arrays discussed in Section 4.7. Some loosening of certain rule restrictions is permissible, provided the planned design does not violate the intention of the rules. Determination of this fact would be based on human judgment, therefore, partially undermining the automatic nature of LSSD. At best, any loosening of these rules would improve circuit/pin count in a relatively minor number of instances.

A better approach would be to enforce strict adherence to the rules allowing violations only upon approval of someone very familiar with the rules intent and the test generation algorithms used. A brief explanation of the intent of each of the rules and the potential relaxation of restrictions which could be allowed is included in the following sections.

The application of LSSD design rules to second level packages in the near future for the MIL environment is less straight forward than that for chip design. The primary problem is the lack of LSSD compatible parts available from the vendors. The first recommendation is that vendors be strongly encouraged to design all future parts, especially LSI and VLSI, to conform with LSSD rules. Additionally, these vendors should be encouraged to redesign some of the more popular MSI/LSI parts to conform with LSSD rules. However until a strong demand from the commercial customers for LSSD materializes, the vendors are unlikely to give much credence to military demands.

In the meantime, the potential exists to apply LSSD concepts to presently available vendor type logic. The feasibility and limitations of this approach should be examined in much more detail in follow on activity for this contract. Even with this type of approach, the testing of LSI/VLSI vendor products which are not LSSD will probably prove difficult.

4.1 LSSD RULE 1

All internal storage is implemented in hazard-free polarity hold latches (i.e., latches which are controlled by clock signals such that the data stored in the latches cannot be changed by other inputs when the clocks are "off").

This rule assures a simple, clocked, synchronous design. Latches are set or reset depending on specific logic levels and not during the rise or fall of a logic signal. It allows latches to be isolated from each other by turning all clocks "off". This is necessary in scanning-in and scanning-out for testing and diagnostics. By this rule, DFF's are not allowed in LSSD.

This rule combined with the following clock rules assures that the generated tests will work as anticipated. Potential fast path problems are completely eliminated. For chip design, this guarantee is very important since access and analysis of internal nodes is virtually impossible.

Conceptually, edge-triggered DFF's arranged in a shift register could accomplish the same result, but require very careful design and wire routing to assure that no fast path problems are encountered, especially for the scan function. For this reason, the use of DFF's is strongly discouraged for chip design. A second level package using vendor components might be able to get away with using DFF's since most nodes can be observed directly and corrections can easily be made to the design. An analysis of this approach should be provided in follow on activity for this contract.

4.2 LSSD RULE 2

The latches are controlled by two or more non-overlapping clocks such that:

- (a) A latch, X, may feed the data port of another latch, Y, if and only if the clock that sets the data into latch Y does not clock latch X.

- (b) A latch, X, may gate a clock C_i to produce a gated clock C_{ig} which drives another latch, Y, if and only if clock C_{ig}' does not clock latch X, where C_{ig}' is any clock derived from C_i .

This rule provides a network with the following properties:

- (a) When a clock is turned "on" during system operation or during testing, neither the data being stored in latches nor the gates of this clock may change. This is especially important to the combinational logic test generators and to the logic partitioning process which must depend upon this rule to prevent the inputs to the combinational logic from changing while the output values are being stored in latches.
- (b) The operation of the system clocks in a non-overlapping mode results in only a maximum delay dependency, i.e., fast circuits cannot cause a system malfunction. If the clocks are overlapped, then the delays through the combinational networks, N_1 and N_2 , must each be greater than the amount of overlap.
- (c) The operation of the shift clocks in a non-overlapping manner guarantees proper operation of the shift registers.
- (d) By preventing a clock from being gated by a latch that it controls, this rule guarantees clean clock pulses that are free of hazards.

Rule 2a would be dropped for "LSSD" using DFF's, since the edge-triggered nature of DFF's would eliminate the need for this rule. However, whenever Rule 2a is violated in such a design, the potential for fast path problems would exist.

Rule 2b could at times be violated if the circuit causing the violation can be gated off while still allowing primary input control of the clock. However, the circuit will more than likely introduce some faults which are untestable using standard LSSD test algorithms. In the majority of cases, the best course is to obey this rule.

4.3 LSSD RULE 3

It must be possible to identify a set of clock primary inputs from which the clock inputs to SRLs are controlled either through simple powering trees or through logic that is gated by SRLs and/or non-clock primary inputs. Given this structure, the following rules must hold:

- (a) All clock inputs to all SRLs must be at their "off" states when all clock primary inputs are held to their "off" states.
- (b) The clock signal that appears at any clock input of an SRL must be controllable from one or more clock PIs such that it is possible to set the clock input of the SRL to an "on" state by turning any one of the corresponding clock PIs to its "on" state and also setting the required gating conditions from SRLs and/or non-clock PIs.
- (c) No clock can be ANDed with either the true or complement value of another clock.

This rule defines the structure for the clocks in any LSSD system. A fundamental fact is that it must be possible to keep all clocks off when all the clock PIs are kept off. Just as important, it emphasizes that any SRL clock input, regardless of how many clock PIs may control it, can be turned on and off from a single clock PI at any given time - hence the independence of clocks. Given this rule, one needs to toggle a single clock PI for a DC test.

This rule is necessary to allow the test generators to have complete control of the clocks. These rules would still be applicable in an "LSSD" design using DFF's. In certain rare instances 3c can be violated if turning on the two clocks simultaneously can be arranged so that any potential race conditions can be gated out.

Most of the time though, creatively defining the clocks to the test generator is sufficient to satisfy the rule and still perform the function. This is because the test generators only recognize as clocks those inputs so defined. The inputs to be used as functional clocks do not have to be defined as the test clocks. A case in point is the on chip clock generator described in Section 3.1.3.2.

4.4 LSSD RULE 4

Clock primary inputs may not feed the data inputs to latches either directly or through combinational logic, but may only feed the clock input to the latches or primary outputs.

If a clock signal is mixed with logic signals and applied to the data input to a latch, a race will usually result since both the data input to the latch and the clock input to the latch will be changing together. If this situation does not exist, under the assumption that only one clock at a time is turned on (except for the A and B clocks which can be on simultaneously) this rule could be violated. Again, this rule can be satisfied by creative clock definition.

4.5 LSSD RULE 5

All SRLs must be interconnected into one or more shift registers, each of which has an input, an output and shift clocks available at the terminals of the package.

This rule is the central concept of LSSD in that it provides all system latches with the ability to scan-in and scan-out. This makes the SRL appear like a primary input and a primary output. It is for this reason that they are referred to as pseudo-primary inputs and pseudo-primary outputs. The use of the SRL macro provides the test generation system with all the necessary information to generate tests without requiring the designer to add a host of special data into the EDS System.

This rule requires the ability to shift all of the shift registers simultaneously. This will tend to reduce the test data volume on a dense product. Of course, to shift in parallel each shift register must have its own unique scan-in and scan-out ports. The A and B shift clocks may be shared among several shift registers, each shift register may have its own shift clocks or a single shift register may have multiple shift A or shift B clocks as desired.

If the test data volume is of less concern than I/O count, multiplexed scan-in's and scan-out's could be allowed. Since the scan-in and scan-out pins can normally serve as functional pins also, a tight pin count due to the number of scan pins is rarely encountered. For this reason, simultaneous loading of shift registers is emphasized.

4.6 LSSD RULE 6

There must exist some primary input sensing condition (referred to as the scan state) such that:

- (a) Each SRL or scan-out PO is a function of only the single preceeding SRL or scan-in PI in its shift register during the shifting operation.
- (b) All clocks except the shift clocks are held "off" at the SRL inputs, and
- (c) Any shift clock to an SRL may be turned "on" and "off" by changing the corresponding clock primary input for each clock.

The sensitizing condition required by this rule guarantees, irrespective of the logic that may exist in the scan path, the continuity of the shift register string of SRLs from scan-in to scan-out such that during a shift operation, each shift register is controlled by its shift clocks and scan-in port only. If this sensitizing condition was controlled from latch outputs, instead of primary inputs (as the rule requires), then during the

shifting process, when all registers are shifting together, the scan state could not be guaranteed throughout the shifting process and thus the shift register could not be loaded or unloaded. This rule should be applied in all cases.

4.7 LSSD RULES FOR TESTING INTERNAL ARRAYS

With the advent of larger circuit densities, more and more LSI/VLSI components are appearing with on chip ROM and/or RAM. As these array structures become increasingly embedded on LSI/VLSI chips, proper testing of them becomes difficult. Redesigning these arrays to conform with LSSD rules would double or triple the chip real estate they require. Therefore, special LSSD rules to handle embedded arrays have been developed and described in recent literature*. Excerpts from that paper are included here.

The arrays to be considered are assumed to be either Read Only Memory (ROM) or Random Access Memory (RAM) and that they will function correctly at any speed slower than the maximum specified. Also, the RAM is assumed to contain N words of M bits each, and that it has M data inputs and M data outputs.

If an array were included within an LSSD design, it would result in the general structure shown in Figure 4.7-1. In order to test such a network, the array operation must at least be controlled from the primary inputs (PI's) of the package. This must be done so that data can be scanned in and out of the SRLs without disturbing the data stored in the array. Furthermore, reading and writing the array must be possible without changing the state of the SRLs. Also, both the test pattern generator and the fault simulator must be able to properly handle the array.

* "A Logic Design Structure for Testing Internal Arrays" E.B. Eichelburger, T. W. Williams, E. J. Muehldorf, R. G. Walther. 3rd USA-JAPAN Computer Conference 1978 pp 14-4-1 to 7.



Conceiveably, developing a "smart" test pattern generator that would generate tests for this type of structure is possible. It would, however, be difficult to develop, would probably have very long run times, and would generate large numbers of tests. Consequently, some additional structure to further simplify this problem is desirable.

Three different problems should be considered:

1. To test some faults in the combinational logic, setting the outputs of the array to certain binary values may be necessary. This will require generating these values at the array inputs and then performing a "write" followed by a "read."
2. Testing other faults in the combinational logic, may require that the effect of the fault be propagated through the array through the remaining logic to either P.O.s or SRLs.
3. To fully test the array the ability to write and read "1" and a "0" in every memory cell must be possible.

Solving any of these three problems is far more complicated than the problem of generating tests for combinational logic.

One alternative structure that seems to solve these problems with a minimum of cost overhead is to design the combinational logic such that a one-to-one correspondence can be established between the array inputs and SRLs or PIs. This will allow any pattern to be easily written into any location in the array. If a similar one-to-one correspondence can also be established between the array outputs and SRLs or P.O.s, then any word in the array can be easily read and observed.

The cost overhead for this ability is generally quite small since most arrays have both a data register and an address register associated with it. Thus, setting up the one-to-one correspondence usually only involves turning "on" the appropriate gating logic.

This additional structure for buried arrays (ROM or RAM) can be summarized as follows:

1. The ability to establish a PI state that will prevent writing into the array during the shifting of data through the SRLs and during the operation of the system clocks must be possible. This PI state will be called the "Array Stable State."
2. Clock signals controlling SRLs feeding array inputs, clock signals controlling SRLs fed by array outputs, and clocks possibly used internally to the array must be controllable from PIs such that sequentially raising and lowering these clocks will result in race-free operation.
3. For each array an "Array Write State" and an "Array Write Sequence" which will establish a one-to-one correspondence between the array inputs (data & address) and SRLs or PIs, and will result in writing into the array must exist. The "Array Write State" is a set of values for PIs or SRLs. The "Array Write Sequence" is a sequence of PI changes.
4. For each array an "Array Read State" which will establish a one-to-one correspondence between the array data outputs and SRLs or P.O.s, and array address inputs and SRLs or PIs must exist. There must also exist an "Array Read Sequence" that will result in reading the array and presenting the array outputs to the corresponding P.O.s or storing them in the corresponding SRLs. The "Array Read State" is a set of values for PIs and/or SRLs. The "Array Read Sequence" is a sequence of PI changes.

The design structure does not provide a way of testing the array at machine speed. It is, however, possible to do a limited amount of AC or delay testing with this structure. Depending on the tester and the array design, it may be possible to specify timing for the "Array Write Sequence" and the "Array Read Sequence." This could test for many of the possible

"AC defects" that could occur in the array. It is also possible to time the changing of the array data inputs and address inputs by either changing the corresponding PI or by timing the last shift clock driving the SRL.

At present, conformance to these rules are the responsibility of the designer. However, the increased testability of the array with a relatively small overhead justifies adoption of these additional LSSD rules for both chip design and second level packages. The impact of these rules on second level designs using vendor logic will be studied in the follow on activity for this contract.

4.8 CONCLUSIONS

The LSSD rules just described, including those for internal arrays, seem to fit military requirements for design testability as they are written. Any violation of these rules should only be allowed for well justified reasons and with an understanding of the potential testability impacts. Conformance to the rules as stated can usually be achieved with relatively small overhead in circuit and pin count provided the logic designer uses his creative abilities.

These rules can immediately be applied to all future LSI/VLSI designs, but, at present, are unfeasible for second level package design using available vendor components. The recommendation for these second level packages is that rules 1 and 2a be relaxed to allow the use of shift registers made of DFF's. The impact and feasibility of this recommendation should be examined in detail in follow on activity for this contract.

SECTION 5

TESTER CHARACTERISTICS TO ACCOMMODATE LSSD

Requirements for testers that apply LSSD test patterns are different from testers that apply non-LSSD test patterns due to the cyclic nature of tests produced by LSSD test generation.

5.1 LSSD EFFECTS ON STANDARD TEST DATA VOLUME

LSSD produces test sets which are extremely large if put in either of the two basic pre-existing data formats.

Bit Organized Data

This format requires one bit of data for each tester pin and, if any one bit is to be changed, all bits must be reapplied. This type of format provides the lowest data volume if tests are structured with many input changes and many output changes per test.

Two-Byte Format

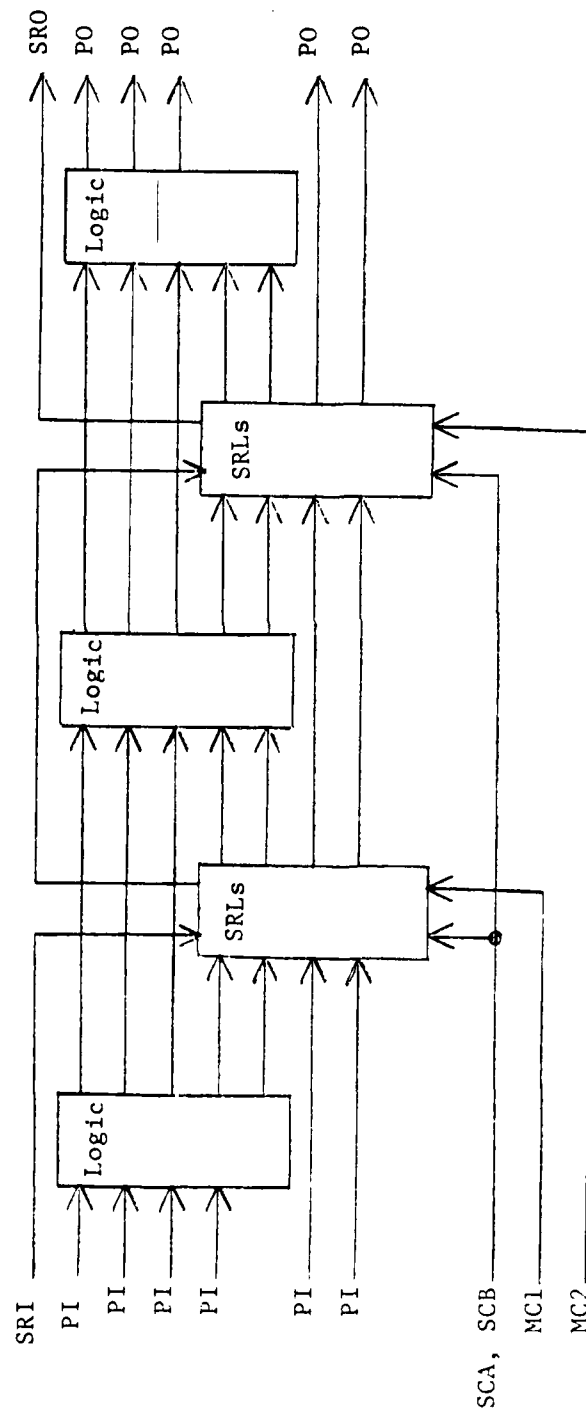
This format uses a 16-bit word to describe each change (input or output) with four to six bits used to describe the function (Force, Expect, 1/0/X) and 10 to 12 bits to describe the tester pin to which that function is to be applied. This type format provides the lowest data volume if tests are structured with a small number of input-output changes per test.

An example of a typical LSSD test will be given to establish a base for LSSD tester descriptions. This is an attempt to describe the general tasks that a LSSD tester will be required to do in a production test.

5.2 TYPICAL LSSD TEST

The following LSSD test is a simplified example that uses only one type and configuration of LSSD latch and one style of an idealized LSSD test. Refer to Figure 5.2-1.

1. Load LSSD SR
 - (a) Apply 1 or 0 on SRI
 - (b) Issue SC A and B
 - (c) Repeat for the full length of shift register
2. Load PI
 - (a) Put the proper states on all primary inputs (many change state).
3. Expect PO
 - (a) Read primary output and check state against expected value.
4. Issue Machine Clocks
 - (a) Issue MC 1 and MC 2.
 - (b) Issue 1 SC B to move normal machine latch contents to their respective L2 latches.
5. Unload Shift Register
 - (a) Read primary output and check state against expected value.
 - (b) Issue SC A and B.
 - (c) Repeat these two steps for the full length of the shift register.
6. The above 5 steps are repeated for each test.



SRI - SHIFT REG IN
 SRO - SHIFT REG OUT
 PI - PRIMARY INPUT
 PO - PRIMARY OUTPUT
 SCA - SHIFT CLOCK A
 SCB - SHIFT CLOCK B
 MC1 - MACHINE CLOCK 1
 MC2 - MACHINE CLOCK 2

Figure 5.2-1. Typical LSSD Product

7. There may be more than one shift register. If there is, Steps 1 and 5 are expanded to put data on and expect data at more than one pin.

The above test is idealized. In the real world, many variations occur to include some random events, additional set up prior to issuing machine and/or shift clocks, inversion in the shift register path, clock inputs used as PI's in some tests, sequential PI changes prior to expects, and arrays.

The total system must be able to handle these kinds of things as well as the idealized test.

5.3 LSSD TESTER HARDWARE

LSSD tester hardware, for the above reasons, provides a set of programmable functions. These functions are basic in nature and, when explained as such, appear to have little meaning as far as the total testing task. This flexibility, however, allows these basic functions to be put together to form larger functions and meet the requirements of testing LSSD products.

5.3.1 RECURSIVE HARDWARE TO ACCOMMODATE LSSD

Testers available in normal use do not have the capabilities listed above. Additional hardware may be added to these universal testers to make them have complete programmability and the capability of handling long serial shift strings. The block diagram of such a tester is shown in Figure 5.3.1-1. Since this additional hardware must handle the cyclic nature of LSSD data, it is named here Recursive.

Recursive is structured into two main sections:

1. Recursive Controller
2. Recursive Memory

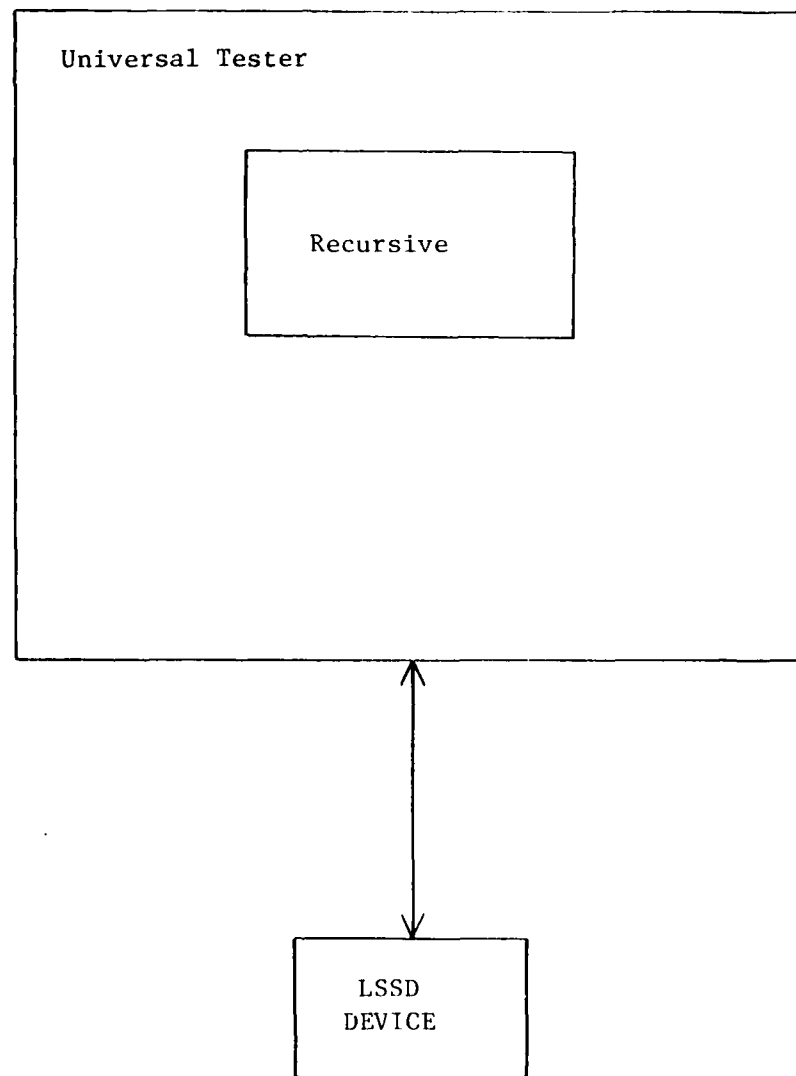


Figure 5.3.1-1. LSSD Tester Hardware

The Recursive Controller is the section which causes Recursive to perform a series of pre-programmed functions. It directs the Recursive Memory section when and how it should function. It also determines when to stop, generate sync, etc.

The Recursive Memory section takes care of obtaining, storing, timing, and decoding the actual test data which is to be applied to the product under test. It also handles eight control functions which are described as part of the test data (different from set up or set up and control functions).

A Recursive block diagram is shown in Figure 5.3.1-2.

5.3.1.1 Recursive Controller

The heart of the Recursive Controller is a small processor made up of:

- (a) 64 x 8 Op Memory
- (b) Op Data Register
- (c) Op Address Counter
- (d) Register Stack (16 28-bit registers)

The Op Memory is loaded with a program of up to 64 8-bit words. The 8-bit words have a format which allows for the following six basic operations:

- (a) Branch - To Op Memory Address XXXX
- (b) Read Stack - Read Stack Register XX
- (c) Stop - Stop
- (d) Reset Counter - Reset Counter 0 or 1 or 2 or 3
- (e) Increment Counter - Increment Counter 0 or 1 or 2 or 3
- (f) Branch If Counter is Not Equal To Its Register - Branch to Op Memory Address XXXX if the previously incremented counter is not equal to its own register; if it is equal, go to the next sequential Op Memory Address.

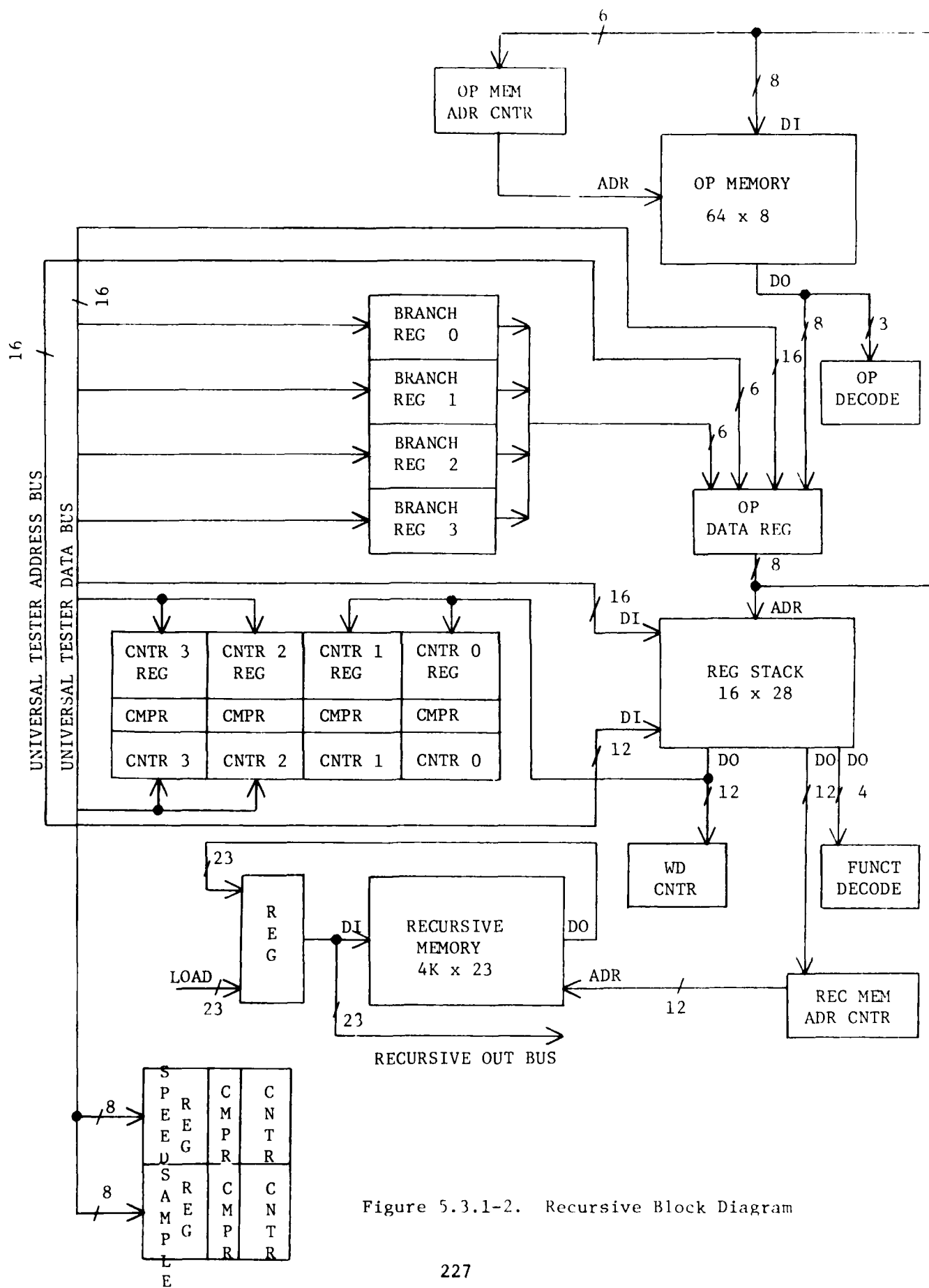


Figure 5.3.1-2. Recursive Block Diagram

These six basic op's are augmented by the Register Stack which is accessed by the Read Stack Op. The Read Stack Op addresses 1 of 16 28-bit registers (Register Stack). Each register contains three fields:

1. Function Field (4 bits)

This field defines the function of the other two fields. There are two types of functions:

- (a) Load Counter Register - This causes the value contained in the Count field to be loaded into Counter 0 or Counter 1 Register (function field determines Counter 0 or 1 Register).
- (b) Recursive Memory Function - Determines which of four types of Recursive Memory Function is to be performed by Recursive Memory. It also directs the Register Stack "Count" field to the Word Counter Register and the "Address" field to the Recursive Memory Address Counter.

2. Count Field (12 bits)

This field contains a count to be loaded into Counter 0 Register or Counter 1 Register, or the Word Counter Register dependent on the Function Field.

3. Address Field (12 bits)

This field contains the Recursive Memory Starting Address for the Recursive Memory Functions defined by the Function Field.

The other five ops are used to provide sequences of Recursive Memory operations and provide status and controlled interrupts back to the universal tester.

Branch Registers 0 Through 3

Recursive can be started using any one of the four branch register contents as the Op Memory Starting Address.

5.3.1.2 Recursive Memory

The Recursive Memory section consists of a 4K x 23 memory, timing, controls and a function counter.

The 4K x 23 memory has 4K 23-bit words. The 23 bits are as follow:

- 16 Bits - Data
- 2 Bits - Data Parity (1 bit per byte)
- 4 Bits - Channel Gate Bits (used to define what the channel word is, i.e. Address, Command, Data, Gate, Recursive)
- 1 Bit - Channel Gate Bit Parity

5.3.1.3 Programmable Timing Of Recursive

Frequently a universal tester will have to slow down individual functions to allow the device under test enough time to stabilize (dependent on technology speed and output drive capability) between forces and samples. However, once Recursive is started, the universal tester can no longer slow down testing. This capability is provided within Recursive by programming Cycle Time and Sample Time counters.

Recursive Cycle Time

Recursive Cycle Time is the length of time between the start of one Memory cycle and the start of the next.

Recursive Sample Time

Recursive Sample Time is the length of time between the start of one Recursive Memory cycle and the sample for that cycle.

Counter 0

This is called the "Cycle Counter" in universal tester programs. It is totally under control of the Op Memory program and is used to count the number of times a Recursive Memory area containing Recursive Data is used during an Op Memory program segment.

Example: (Refer to typical LSSD test, Item 1)

- (a) Apply 1 or 0 on SRI - Would be Recursive Data in Recursive Memory Area 1.
- (b) Issue SC A and B - Would be Recursive Data in Recursive Memory Area 2.
- (c) Repeat for Full Length of Shift Register - Counter 0 circuitry would be used to count how many times Items 1 and 2 were applied. The programmed value would be the length of the Shift Register. When the value was reached, Counter 0 would be equal to Counter 0 Register and the Op Memory program would fall through to Step 2 of the LSSD test.

In Step 2 (Apply PI), all primary input information would be Recursive Data in say Recursive Memory Area 3. Since this area is only used once, Counter 0 Register would be programmed to 1 and, after applying Area 3 once, the program would fall through to Step 3, etc.

Counter 1

This is called the "Loop Counter" by part number programs. Its primary purpose is to allow random tests to be handled in various areas of an LSSD test by a single Op Memory program.

Counter 2

This is called the "Pattern Counter" by part number programs. A LSSD pattern is defined as follows:

- (a) An LSSD Shift Register unload is one pattern (includes many "samples").
- (b) With the exception of Definition 1, each "sample" (test for product output compared to expected value) is a pattern.
- (c) Each test starts with a pattern of 0 (see Counter 3 explanation for a definition of a test).

The "pattern" counter is reset and incremented as part of the Op Memory program.

The pattern counter has two basic uses:

- (a) To keep track of the pattern number being applied to the product under test so that if the product fails, the universal tester can read the pattern number for product diagnosis purposes.
- (b) To provide a programmable stop condition.

Counter 3

This is the same as Counter 2. The only difference is its defined usage as a "Test Counter". A test is defined as a loopable entity and identifies a group of patterns. Each LSSD test is a test (the total typical LSSD test presented is one test). The "Test Counter" is incremented as part of the Op Memory program.

The Test Counter has two basic uses:

- (a) To keep track of the test number being applied to the product under test so that if the product fails, the universal tester can read the test number for product diagnosis purposes.
- (b) To provide a programmable stop condition.

5.3.2 SUMMARY

The tester described above called Recursive can satisfy the requirements of LSSD. That is, Recursive can handle long shift strings, the cyclic nature of LSSD tests, and is completely programmable. Long shift strings can be handled by the 4K x 23 Recursive memory. Counter 0 can be programmed to the length of the shift register string allowing for the variation in hardware designs. Counter 1 allows for random tests to be inserted into an LSSD test, Counter 2 records the pattern number being applied in case of a pattern failure, and Counter 3 records the test being applied. The four branch registers allow for branch on condition depending on data read/compare. The details of Recursive will certainly depend on the type of universal tester being used. But the programmability and ability to handle long serial data strings will be universal among them all.

SECTION 6

LSSD REQUIREMENTS FOR CAD SYSTEMS

There are several features required of the various subsystems of computer aided design (CAD) to accommodate LSSD designs.

6.1 SIMULATION

The simulation subsystem has two requirements that are imposed by the LSSD design philosophy. One is a means to verify that the design constraints have been met. The other is a means to represent the LSSD latch to the system.

6.1.1 DESIGN RULES CHECKING

Prior to the designer performing good machine simulation to verify functionality, the design rules checking subsystem should be exercised. This subsystem is required to verify that the logic design has not violated any rules dictated by the LSSD design philosophy. This is accomplished by generating stimuli under program control that reflects the LSSD design rules. This stimuli is then applied to the design via the simulator and the results are checked to verify that the LSSD design rules have not been violated. Algorithms of this type have been described in recent literature*.

6.1.2 LSSD SRL MODEL

In order to accommodate the design rules checking algorithms special models of the SRL must be developed. These models must be able to break the feedback path of the SRL so that all logic will appear combinational in nature. The models must also be able to act as primary inputs and primary outputs for both test generation and design rules checking. Without these features, the LSSD SRL loses most of its advantages.

* "Automatic Checking of Logic Design Structures for Compliance with Testability Ground Rules" H. C. Godoy, G. B. Franklin, & P. S. Bottorff pp. 469-478. IBID

6.2 TEST PATTERN GENERATION

The test pattern generation subsystem also has several requirements that are imposed by the LSSD design philosophy. The test pattern generator must be capable of identifying controls, automatic pattern generation, pattern subsumption (the technique of reducing the number of test patterns by combining them where don't care states allow), fault simulation, and design partitioning.

6.2.1 IDENTIFICATION OF CONTROLS

The test pattern generator must be able to identify those primary inputs that are system clocks and scan controls as well as the scan path. Without this capability the test pattern generator would not be able to take advantage of the LSSD design structure. The control identification may be made in the logic description in fields that have been identified to the test pattern generator.

6.2.2 AUTO TEST PATTERN GENERATION

The test pattern generator must be capable of handling the shift register type circuits and take advantage of the LSSD architecture when generating test patterns by treating the SRLs as primary inputs and primary outputs. To accomplish this, the simulation models described in 6.1.2 can be used. This capability requirement is similar to the requirements given in Section 6.2.1 in that the test pattern generator must be structured to handle the LSSD designs. On the other hand this requirement may cause the test pattern generator to become somewhat restrictive in that it may not function effectively for non-LSSD designs.

6.2.3 PATTERN SUBSUMPTION

The test pattern generation subsystem should also support pattern subsumption (the technique of reducing the number of test patterns by combining them where don't care states allow). Pattern subsumption is required for LSSD due to the extra time incurred by shifting the test data in and out of the scan string.

6.2.4 FAULT SIMULATOR

A fault simulator should be available to exercise the resulting test patterns in order to evaluate their effectiveness. This fault simulator should meet the same requirements as the good machine simulator. The difference between the simulators is that the fault simulator is capable of producing test faults in the logic description and then simulating the logic using the test patterns as input to verify that the patterns cause the test fault to be produced at an output.

6.2.5 DESIGN PARTITIONING

For improved automatic design partitioning, the SRL's must be identified as potential boundary points in addition to primary inputs and primary outputs. This again can be accomplished using the simulation model described in Section 6.1.2.

6.3 PHYSICAL DESIGN

The physical design subsystem is not greatly affected by the LSSD design philosophy. The auto place and wire programs require a feature that is desirable but not absolutely necessary. Since most auto place and wire programs place circuits by connectivity, the scan path connections may interfere with auto place and wire performance. It is desirable to be able to force the auto place and wire programs to ignore the scan path. This would require that the scan path be manually wired and that the LSSD latch placement would be less than optimal (longer wire lengths). Since the scan path is not operational during system operation, optimal placement is not necessary.

6.4 DATA RELEASE

Data release is the least affected by the LSSD design architecture. There are no additional features required for data release.

SECTION 7

PROJECTION OF LSSD EFFECTS

The effect of imposing LSSD constraints on a design in many cases depend on the design technique normally used. In general, three synchronous design techniques are commonly used by logic designers:

- (a) Edge-Triggered Design
- (b) Master-Slave Latch Design
- (c) Multi-Clock Design

Figures 7.0-1, 7.0-2, and 7.0-3 illustrate the general circuit configuration for each of these design techniques. The edge-triggered design represents the technique most commonly used in military designs. The projected effect of LSSD on converting from any of these design techniques is found in the following subsections.

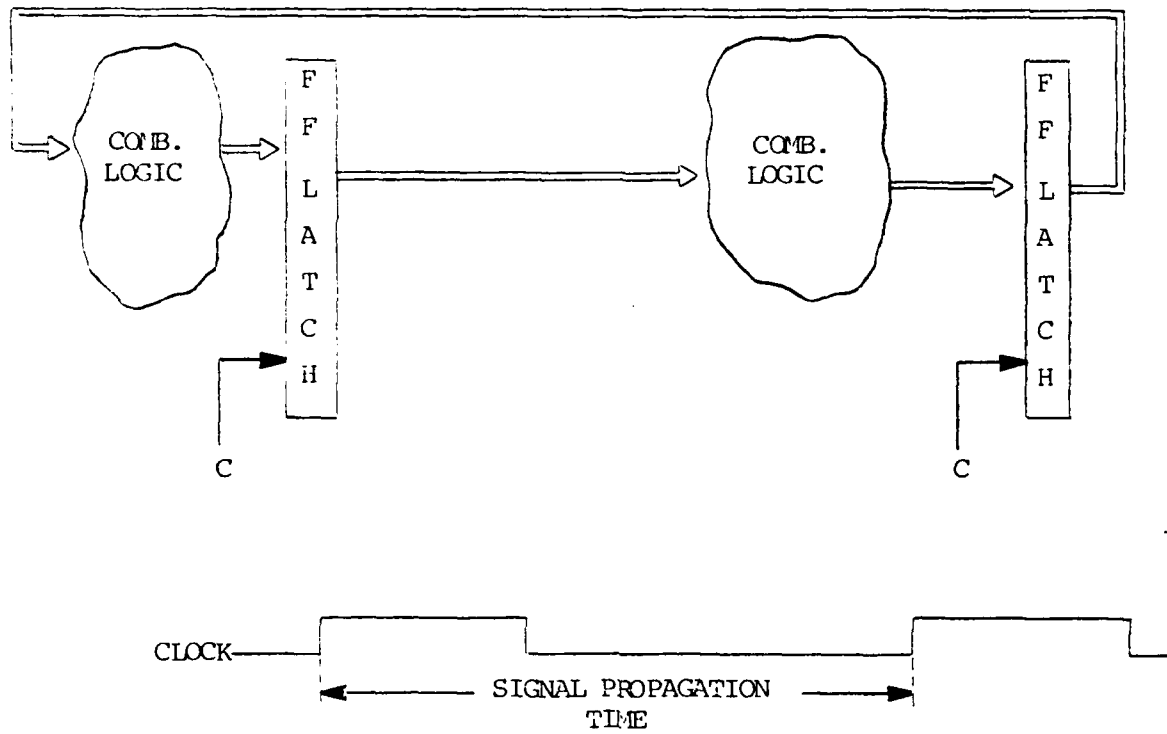
7.1 CIRCUIT DENSITY

To properly analyze the effect of LSSD on circuit density, two separate LSI/VLSI approaches should be considered: semi-custom and custom. A discussion of LSSD implementations in these two technology approaches are included here.

7.1.1 CIRCUIT DENSITY FOR SEMI-CUSTOM DESIGNS

Semi-custom chip designs, represented by master slice approaches, consist of a predetermined number of circuits or cells which are available to the designer. The discrete nature of these chips, allows LSSD overhead equations to be generated. Dividing the number of combinational cells or gates by the number of latches in the design results in a value K which can be used in these LSSD overhead equations.

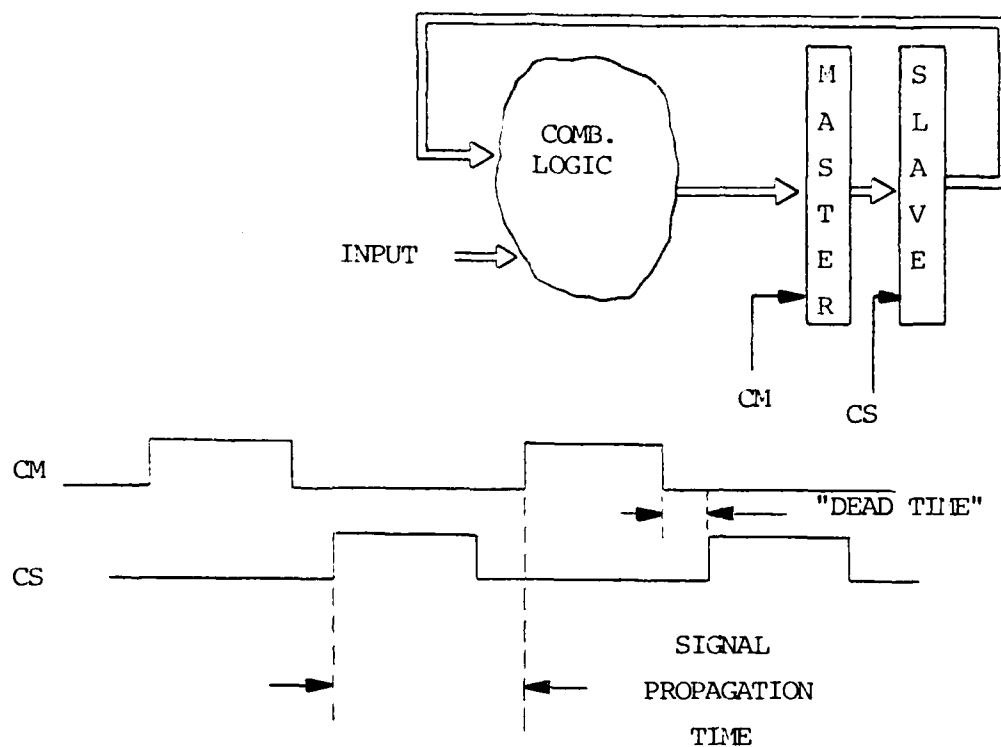
1) EDGE-TRIGGER DESIGN



NOTE: CLOCK DISTRIBUTION SKEW AND FAST PATHS ARE CRITICAL DESIGN AREAS -
NOT LEVEL SENSITIVE

Figure 7.0-1

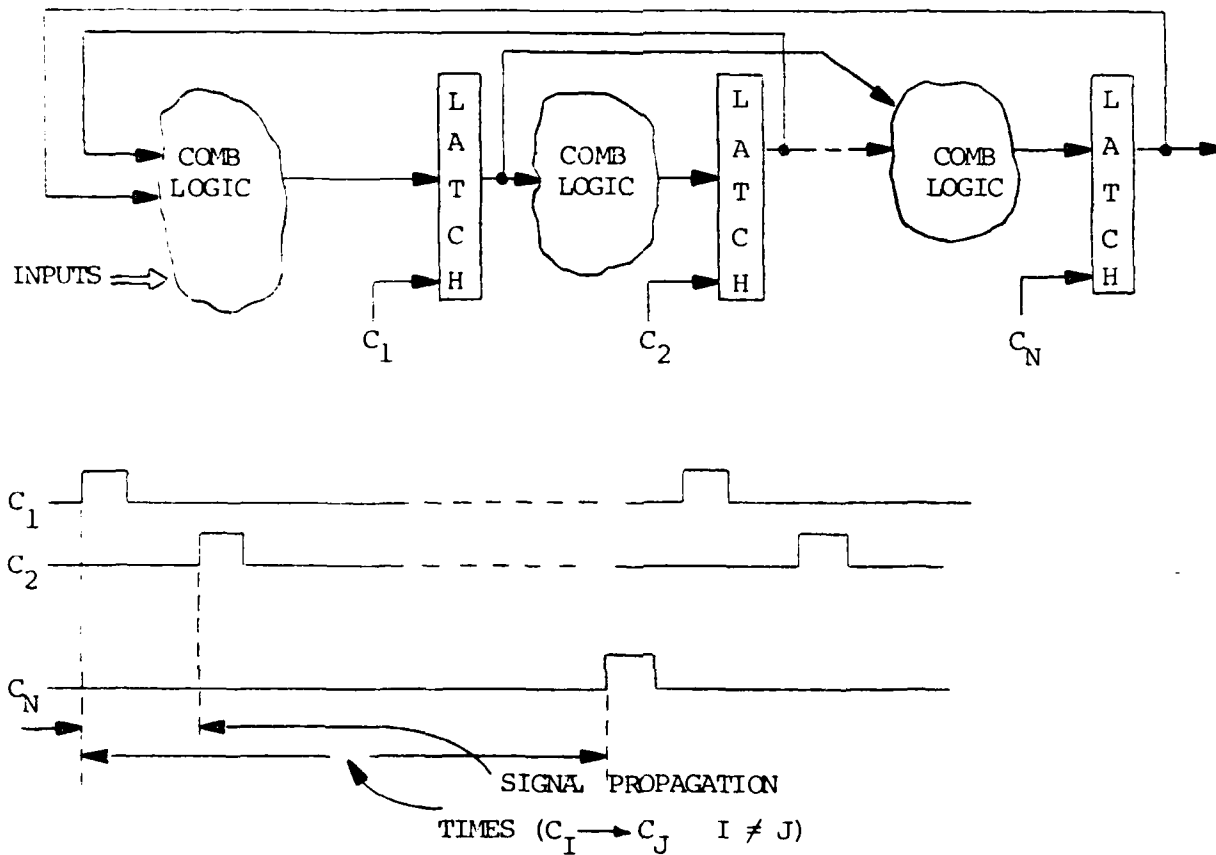
2) MASTER-SLAVE LATCH DESIGN



NOTE: NON-OVERLAPPED CLOCKS PROTECT AGAINST CLOCK DISTRIBUTION SKEW AND FAST PATHS. LEVEL SENSITIVE DESIGN.

Figure 7.0-2

3) MULTI-CLOCK DESIGN



NOTE: LEVEL SENSITIVE DESIGN WITH NO "DEAD TIME" IN THE CLOCK CYCLE

Figure 7.0-3

The number of cells or gates required to implement an edge-triggered DFF(D), a polarity hold latch (P) and an SRL(S) should first be determined for the technology involved. The latch overhead (L) for LSSD can then be determined using the following equations:

Latch Overhead

- (a) Edge-Triggered $S-D = L$
- (b) Master-Slave $S-2P = L$ (except where master-slave can be implemented in less than 2P)
- (c) Multi-Clocked $S-P = L$

For determining clock driver overhead (C), the fanout (F) for the clock drivers must be determined. For technologies which can use the same clock driver for all three designs, the following equations apply:

Clock Driver Overhead

- (a) Edge-Triggered $2/F = C$
- (b) Master-Slave $1/F = C$
- (c) Multi-Clock $2/F = C$

For technologies which would require different clock drivers for DFF's than those for the SRL's and polarity hold latches, the number of cells or gates required for the DFF clock driver (G_d) and for the level sensitive clock driver (G_p) as well as their respective fanouts (F_d) and F_p) should be determined. Then the following equation for edge-triggered designs should be used:

$$\text{Edge-Triggered} - \frac{3 G_p}{F_p} - \frac{G_d}{F_d} = C$$

If a sufficient number of latches exist, additional overhead may be required for LSSD clock driver powering trees. For simplicity's sake, the LSSD effect on powering trees is ignored. The resulting overhead equations tend to represent a minimum LSSD cost anyway, so ignoring this factor should not affect their validity.

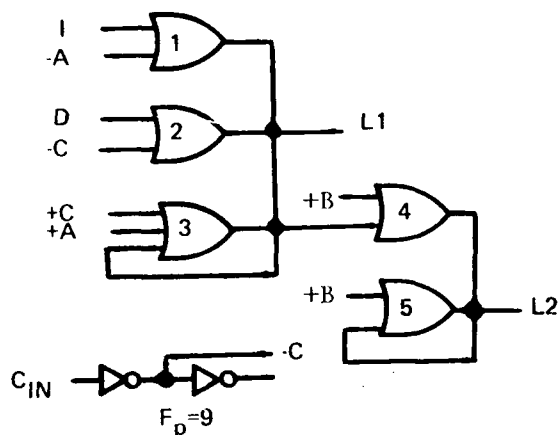
Once the values for L and C have been determined, the percent LSSD overhead as a function of logic blocks per latch, K, can be calculated using the following equations:

% Overhead

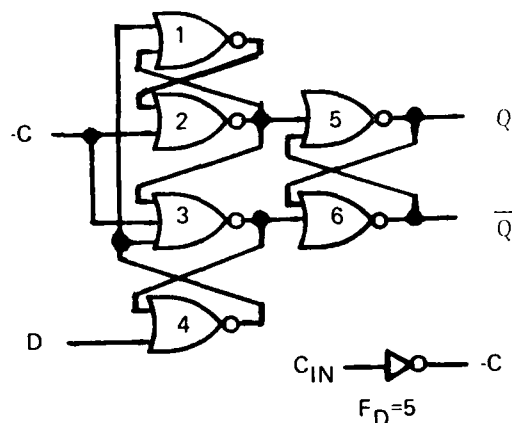
- | | |
|--------------------|-----------------------------|
| (a) Edge-Triggered | $(L+C)/(K+D) \times 100\%$ |
| (b) Master-Slave | $(L+C)/(K+2P) \times 100\%$ |
| (c) Multi-Clock | $(L+C)/(K+P) \times 100\%$ |

An example of applying these equations to MS399 is shown in Figure 7.1.1-1. A graph of these equations for values of K between 0 and 35 is included for all three design techniques in Figure 7.1.1-2. The important thing to note is that the cost for converting an edge-triggered design to LSSD in MS399 is actually negative because an SRL can be implemented in fewer cells than a DFF. The low cost incurred for substituting LSSD for edge-triggered designs is fairly typical for all technologies since the DFF is usually about the same size as the SRL. As an example, the graph for LSSD overhead for converting edge-triggered designs into an IBM TTL masterslice is also shown in Figure 7.1.1-2.

The only significant LSSD overhead costs are associated with the replacement of single polarity hold latches with SRL's. This fact is clearly demonstrated by the fact that the polarity hold latches in the A and B latches were the sole reason that the LSSD overhead was greater than zero in the SCS481.



SRL, PH and Clock Driver



DFF and Clock Driver

1) Edge Triggered:

$$D=6, S=5, G_D=1, G_P=2, F_D=5, F_P=9$$

$$\text{Latch Overhead} = L = 5 - 5 = -1$$

$$\text{Clock Driver Overhead} = C = 3(2/9) - 1/5 = 0.467$$

$$\% \text{ LSSD Overhead} = \frac{-0.533}{K+6} \times 100$$

2) Master Slave Double Latch:

$$P=2, S=5, F=9$$

$$\text{Latch Overhead} = L = 5 - 2(2) = 1 \quad (\text{Block 1})$$

$$\text{Clock Driver Overhead} = C = 1/9 = 0.111$$

$$\% \text{ LSSD Overhead} = \frac{1.111}{K+2(2)} \times 100 = \frac{1.111}{K+4} \times 100$$

3) Multi-Clock Single Latch:

$$P=2, S=5, F=9$$

$$\text{Latch Overhead} = L = 5 - 2 = 3 \quad (\text{Blocks 1, 4, 5})$$

$$\text{Clock Driver Overhead} = C = 2/9 = 0.222$$

$$\% \text{ LSSD Overhead} = \frac{3.222}{K+2} \times 100$$

Figure 7.1.1-1. LSSD Overhead in MS399

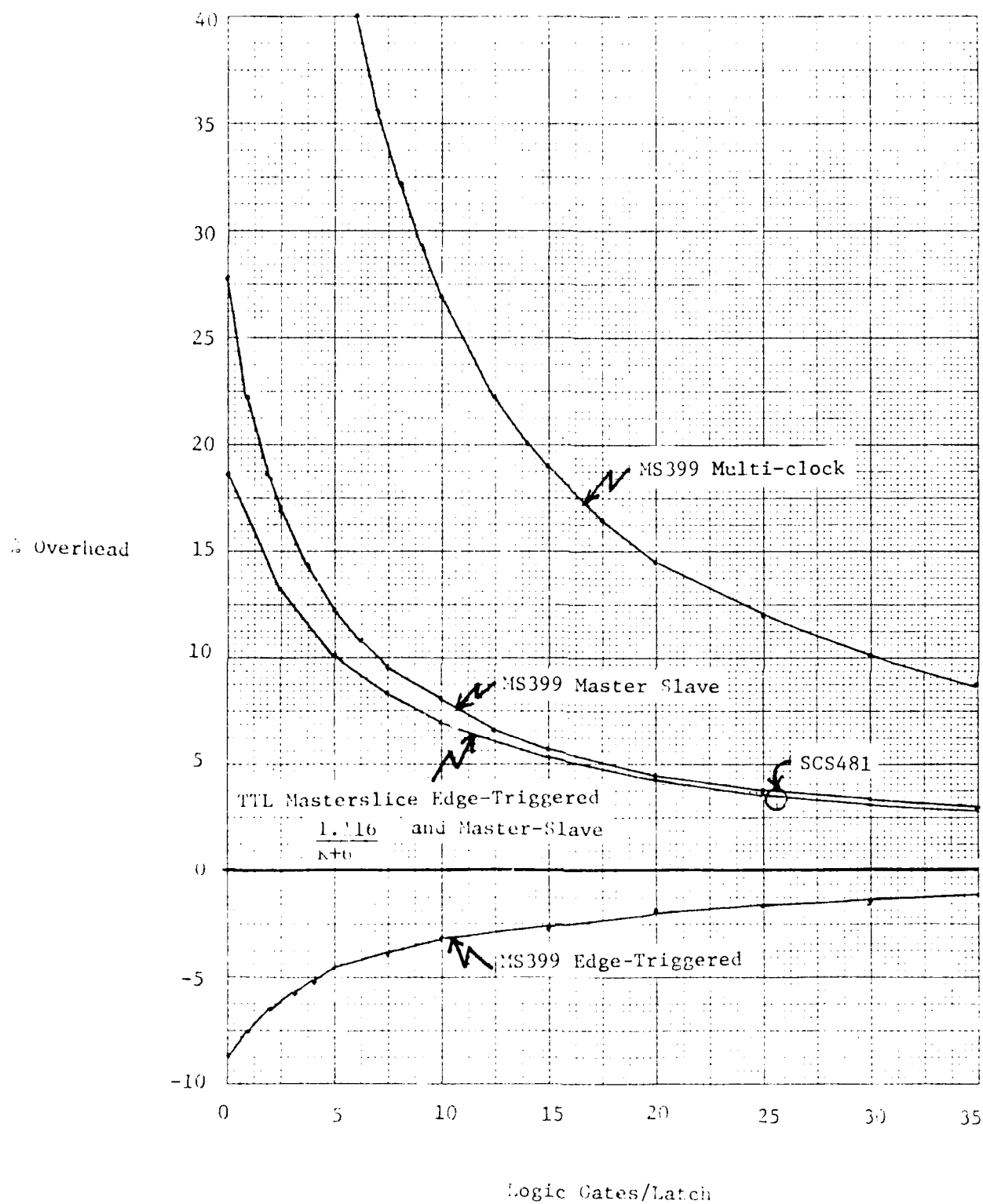


Figure 7.1.1-2 LSSD Overhead

The numbers obtained from the previous equations, generally represent the minimum LSSD cost incurred. Since all clock drivers are not normally used to their fanout limit, and since powering trees are sometimes required, the clock driver overhead estimate tends to be understated. Also any on chip clock generation circuitry will occupy additional space on the chip. However, a clever designer, with a flexible variety of SRL's available, can often reduce the overhead associated with latches, especially in a multi-clock design. The net effect is that the numbers obtained from the equations have tracked well with actual figures from IBM LSSD designs.

7.1.2 CIRCUITRY DENSITY FOR CUSTOM DESIGNS

Custom chip designs, on the other hand, are not constructed from a predetermined number of circuits or cells. This allows the designer the freedom to construct the SRL circuits as well as the standard logic circuits in a minimum area. In this case the LSSD overhead is not as easily defined as in the semi-custom case. The physical designer's implementation of the LSSD latch as well as variables introduced by varying the technology that might be used can have a significant effect on the results of the computations made to determine the LSSD overhead required for the total design. The results of Sec. 7.1.1 can be used directly to approximate the LSSD overhead. Determining the actual overhead can be accomplished only after the design is completed using a similar derivation as in sec. 7.1.1 but for this case the factors would be computed using circuit areas.

7.2 PINOUT REQUIREMENTS

The pin overhead for LSSD again depends on the design technique being replaced. Two pins which are always required are the scan in and scan out pins. The scan in pin can be shared with any non-clock or non-scan control primary input. The selection of the scan in pin should be made with careful consideration of the integration of the device into the second level package. In almost all cases, a separate scan in pin is not required.

Likewise, the scan out pin can be shared with any primary output pin which feeds directly from an SRL or which feeds from an SRL through combinational logic which can be gated out from primary inputs. An example of this would be the data registers feeding through the data out port mux in the SCS481. Again, the selection of the scan out pin should take into account the second level implementation.

A separate A clock pin for shifting is required no matter what design technique is being replaced. The A clock pin can be shared with some other primary input, but another primary input which gates off the non-A clock function of that pin would be required. The net result is that an additional pin is required.

The B clock pin is the same as the slave clock pin for a master-slave design. Therefore, a master-slave design does not require an additional pin for 16 B Clock. However, for edge-triggered and multi-clock designs, the B clock represents an additional pin required for LSSD.

For simple clocking schemes, the number of system clock pins required for LSSD are the same as those for the non-LSSD designs. If the clocks are generated on chip, one primary input pin per generated system clock is generally required. For complicated clock generation circuits, this can significantly add to the LSSD pin count.

In general, the LSSD pin overhead minimum for the master-slave design is one pin for the A clock. For an edge-triggered design, the LSSD pin overhead minimum is two pins for the A and B clocks. A multi-clock design has a minimum LSSD pin overhead of two pins for the A and B clocks and one

pin per on chip generated system clock. These overheads are summarized below:

LSSD Pin Overhead

(a) Edge-Triggered Design	2 pins
(b) Master-Slave Design	1 pin
(c) Multi-Clock Design	2 pins + 1 per/generated system clock

7.3 LAYOUT COMPLEXITY

Layout complexity is increased somewhat by the use of LSSD. The increase in layout complexity is a result of the addition of the extra clock lines that are common to all of the latches in the design and the creation of the scan path linking each of the latches to the next. This effect is considered minimal. The scan path wiring has no effect on performance so these nets may be assigned the lowest priority to the auto place and wire programs resulting in longer wire lengths. The LSSD clock wiring is a little more restrictive. When the wire lengths increase for the clocks, the increased loading may require higher power circuits or the addition of clock driver circuits to the clock driver powering trees, resulting in additional circuit area.

7.4 PERFORMANCE LIMITATIONS

LSSD effects on performance, again, are design technique dependent. For a master-slave or multi-clock design, the performance impact is basically zero. The L1/L2 configuration of the SRL is identical to the master-slave latch configuration so no speed impact is apparent. In a multi-clock design, the L1 latch of the SRL replaces the normal multi-clock polarity hold latch. The L2 latch is not placed in the signal propagation path, so the only performance impact it introduces is the slight increase in the L1 propagation due to the additional fanout to the L2.

The LSSD performance effects on edge-triggered designs are less obvious. The elementary gate representations of the two most popular versions of the basic shift register latch (SRL), the polarity hold (PH) SRL, which is the version used in the SCS481, and the clocked set/reset (CSR) SRL, are shown in Figures 7.4-1 and 7.4-2 respectively. A NOR gate representation of an edge-triggered delay flip-flop (DFF) is shown in Figure 7.4-3.

Using the timing diagrams in Figure 7.4-4, an analysis of the operational speed of the latches can be performed. For all three latches the data input must be valid some time S before the clock rises. Since the C Clock is high for the SRL's, the data can pass through the latch to L1. Upon the rising edge of the clock, data begins propagating through the DFF, taking time PD, and the C Clock is turned off latching the data into L1. Some time P later, determined by the clock generation circuitry, the B Clock is turned on to the SRL's, starting data propagation through the latch to L2, measured by time T.

The values for S, P, T, and PD, as calculated for each latch in unit gate delays, are shown in Table 7.4-5. The elementary logic representations show more 'gate' delays than indicated in the table because they do not accurately represent actual circuit implementation in the technology. By the judicious use of dotting or function extension, i.e., wired-or and gate width (more inverters on the same load) techniques a unit gate can perform several elementary logic functions. These techniques were used to generate the actual path delays shown.

The time P for the SRL's is shown to vary from 0 to 2 unit gate delays. This time is under complete control off chip and can be thought of as a latch 'speed control'. P can be varied to guarantee that a fast path through a latch does not cause the data input of another latch to change during the hold time of that latch, causing the incorrect value to be stored. As the minimum combinational logic distance between latches in the design increases, P can be decreased until it reaches zero.

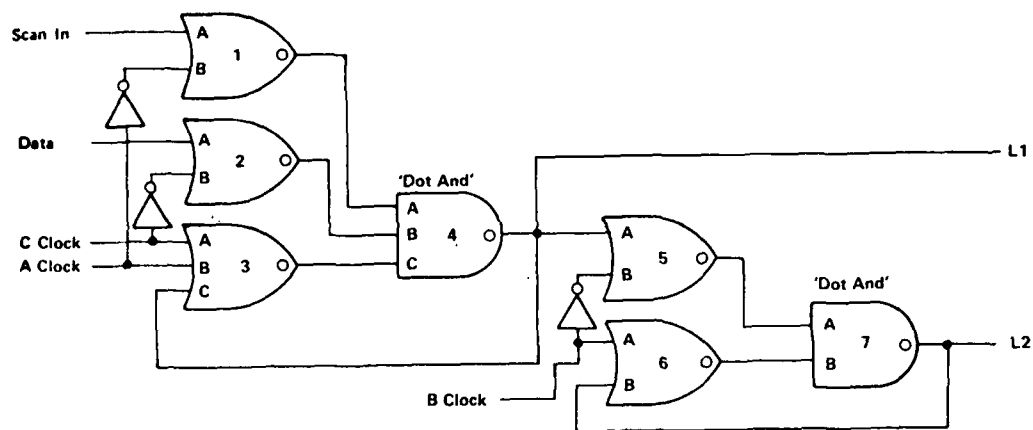


Figure 7.4-1 Polarity Hold (PH) SRL

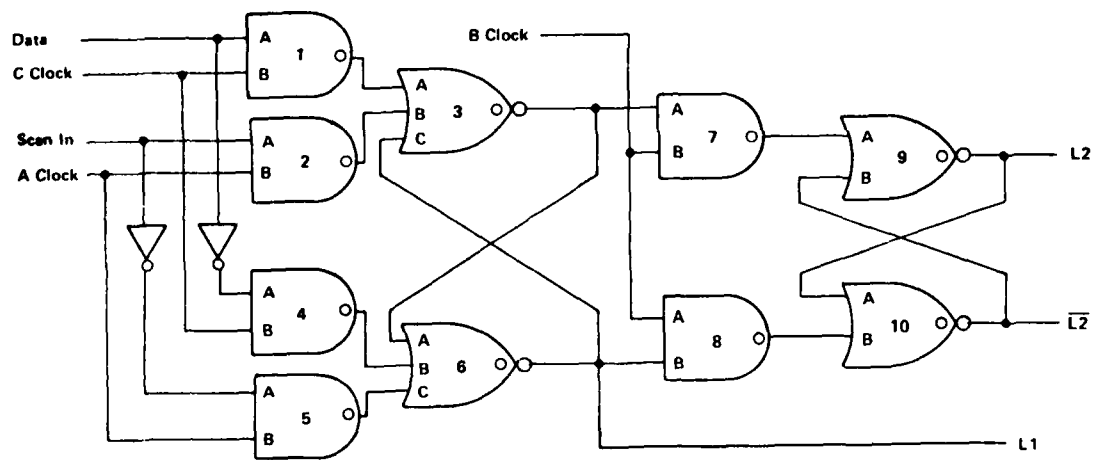


Figure 7.4-2 Clocked Set/Reset (CSR) SRL

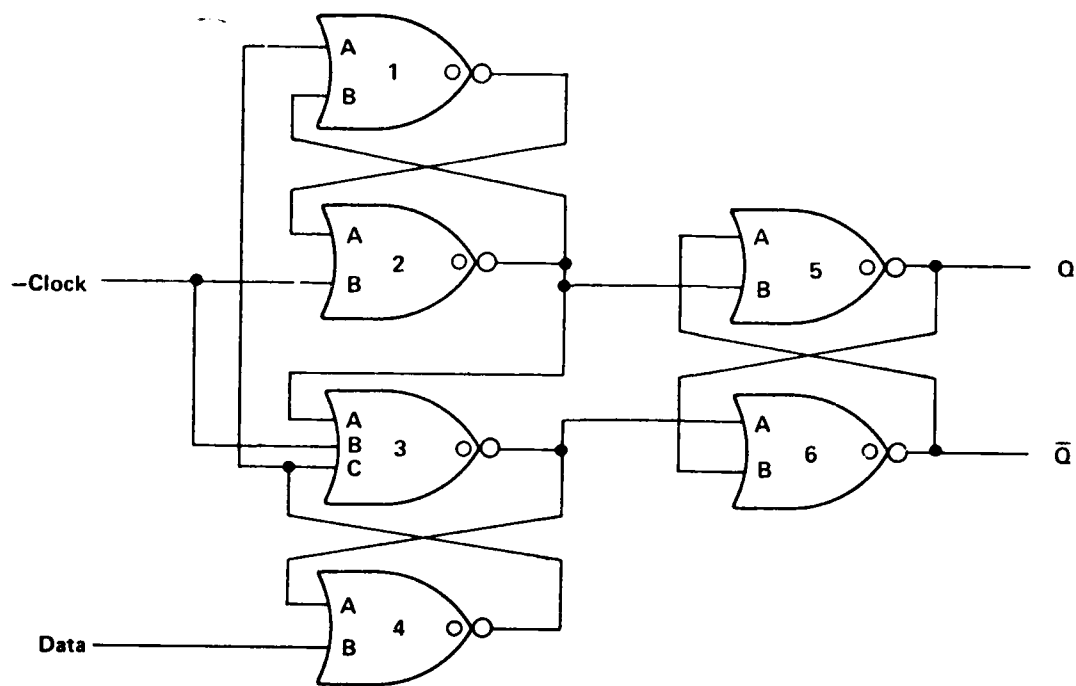


Figure 7.4-3 DFF#1

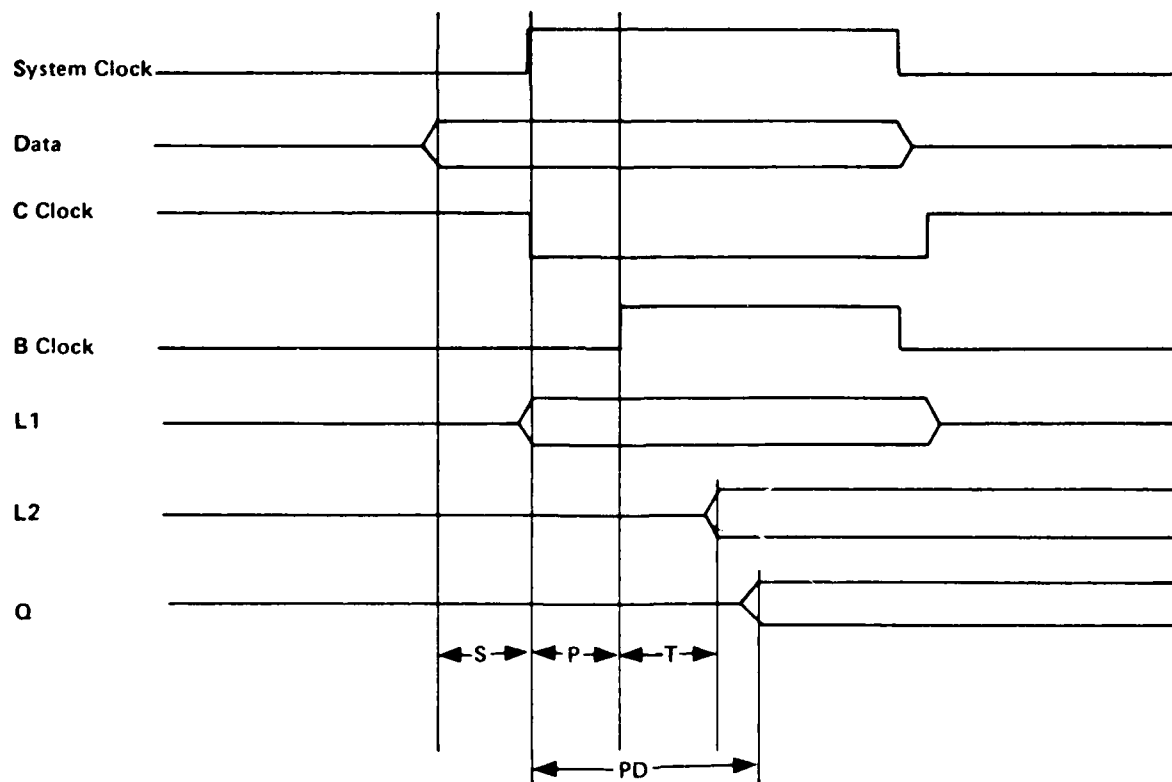


Figure 7.4-4 Latch Timing

Table 7.4-5 SRL vs DFF Performance Comparison

SRL - Polarity Hold

Time From Stable Data On Input Of L1 To Stable Data On Output Of L2:

- 1) Set Up Time For L1 Input = 1 Gate Delay (S).
- 2) Clock Time Between C And B \emptyset To 2 Gate Delays (P).
- 3) L2 Input To Output Propagation Delay = 2 Gate Delay (T).

SRL - Clocked Set/Reset

Time From Stable Data On Input Of L1 To Stable Data On Output Of L2:

- 1) Set Up Time For L1 Input = 3 Gate Delay (S).
- 2) Clock Time Between C And B \emptyset To 2 Gate Delays (P).
- 3) L2 Input To Output Propagation Delay = 2 Gate Delay (T).

DFF

Same Analysis Using DFF (Edge Sensitive):

- 1) Set Up Time = 2 Gate Delays
- 2) Clock To Output = 3 Gate Delays (PD).

CONCLUSION: No Significant Performance Advantage For DFF. DFF Is More Complex Design. Edge Sensitive Designs Add Clock Skew Problems, and fast path problems.

Totals (Set Up and Propagation Delay):

Polarity Hold SRL	3-5 Gates
Clocked Set/Reset SRL	5-7 Gates
Edge Triggered DFF	5 Gates

For edge-triggered designs, such fast path problems must be corrected by additional design passes; and, if this error masks other errors, even more design passes may be required. Automatic test generators cannot anticipate these fast path problems either; and, therefore, test generation become more difficult. Level sensitive design allows the latch speed to be adjusted after chip manufacturing, many times reducing the number of design passes required and simplifying the test generation problem.

Comparing total latch delay, set up time plus propagation delay ($S+PD$ or $S+P+T$), at the bottom of Table 7.4-5, the DFF does not appear to have any significant speed advantage over the SRL's. The speed of the SRL can be further improved by using the L1 as the only storage device as in a multi-clock design. This decreases the propagation delay to T with virtually no set up time. Clock generation and timing for this type of circuit is more demanding, but the speed increase may be worth it. In this application the L2 latch often goes unused except for test purposes.

In summary, the LSSD performance impact on all three design techniques in most cases is negligible. The only exception is for an edge-triggered design with less than two or three gates of combinational logic between latches such as in a shift register. When the inter latch combinational depth is this low, the value of P must be increased slowing down the overall propagation speed. However, edge-triggered designs with low combination depth between latches are the most subject to fast path problems which can be avoided with LSSD.

7.5 POWER CONSUMPTION

The effects of LSSD on power consumption is directly related to LSSD effects on circuit density which was discussed in Section 7.1. Power consumption for an LSSD design should consume the same percent increase in power as the percent increase in gates or surface area required for LSSD. Therefore, the same equations and percentages discussed in Section 7.1 should also be applicable to power consumption calculations.

7.6 LOGIC TESTER COMPLEXITY

An LSSD design does not necessarily require any changes to existing testers. The only problem is that LSSD patterns tend to greatly increase test data volume and tester time if the standard tester is not enhanced to handle LSSD patterns specifically. An in depth description of the kind of hardware which must be added to standard testers to accomodate LSSD is provided in Section 5.

7.7 TEST PATTERN GENERATION COST

In general, the effect of LSSD on test pattern generation cost is to significantly reduce those costs when compared to other approaches to test generation. LSSD test pattern generation is more efficient than both manual test pattern generation and non-LSSD automatic test pattern generation. Some of the features of LSSD which allow for this decrease in test pattern generation cost are described in the following subsections.

7.7.1 TEST PATTERN INTEGRITY

The fact that LSSD designs involve the use of race free logic guarantees that the test patterns generated will function exactly as anticipated. The speed of the logic tester used is unimportant, provided that it does not operate faster than the target technology.

In an unstructured non-LSSD design, the elimination of fast paths and control of race conditions are mostly at the discretion of the logic designer. Any errors by the logic designer or improper delay modeling of the logic can cause the test patterns generated for the design to function differently in the physically implemented device. Problems of this nature require detailed analysis to determine the cause of the errors, as well as, regeneration of the defective patterns. LSSD eliminates the need for this effort.

LSSD test patterns also have the advantage of being independent. Any test loop which begins with a scan in and ends with a scan out can be executed independently. Many non-LSSD patterns are dependent upon previous results stored in the memory elements by previous patterns. The independence of LSSD tests allows tests to be skipped or to be run separately to enhance any debug operations being performed.

7.7.2 COMBINATIONAL LOGIC TESTING

The shift register strings incorporated in LSSD designs simplifies testing to a strictly combination level. The SRL's can all be tested using a predetermined test sequence called a shift register test (Section 3.9.4). All remaining faults are basically combinational in nature. To further enhance the testability of this combinational logic, each SRL surrounding this logic can be treated as a primary input or primary output by the test generator because any value can be scanned in or scanned out for input or observation. The addition of all these internal test points, as well as, the elimination of all sequential testing greatly simplifies the job of the automatic test generator resulting in significantly lower test generation costs.

7.7.3 AUTOMATIC PATTERN SUBSUMING

Pattern subsuming involves combining several individual patterns which were generated for a single fault into a single, one more comprehensive pattern. Patterns can be subsumed provided that no primary input

value conflicts exist between the various patterns. Since the outputs from a combinational logic network is independent of the order in which its inputs change, pattern subsuming for combinational logic is perfectly acceptable.

An LSSD design reduces the test problem to a strictly combinational level. This means that test patterns can automatically be subsumed by a test generation program without any concern for races or sequential effects.

Non-LSSD test patterns can be subsumed manually, if done very carefully, or possibly could be done automatically if a sufficiently intelligent test generation program was available. However, any incorrectly subsumed non-LSSD patterns will cause the very problems discussed in Section 7.7.1 resulting in even more manual effort. LSSD completely eliminates the possibility of such problems while reducing test data volume and fault simulation time by allowing for automatic pattern subsuming.

7.7.4 SIMPLIFIED PARTITIONING

Since SRL's can be treated as primary inputs or primary outputs, they provide an additional boundary for partitioning a design. By partitioning a design into little combinational sections between SRL's and primary inputs, primary outputs, or other SRLs, test generation and fault simulation time can be greatly reduced, especially for very large designs.

Partitioning also reduces the cost of generating new test patterns for engineering changes (EC). In an unpartitioned design, EC's may require complete regeneration of test patterns. For a partitioned design, only the affected partitions are required to have new patterns generated.

7.7.5 SECOND LEVEL APPLICATIONS

Provided that a chip has been properly partitioned and can be easily isolated, the same test patterns used at the manufacturing level can be used to test the device in higher packaging levels. The proper isolation and partitioning is more easily accomplished with LSSD because of the additional boundaries created by the SRL's.

Also the reduced number of patterns generated by LSSD pattern sub-summation becomes significant when these patterns are to be stored at the system level for PM/FL type applications. A large pattern set for each LSI/VLSI in the system would rapidly entail very large memory requirements to store them. LSSD can help reduce this requirement.

7.7.6 CONCLUSIONS

LSSD provides several advantages which significantly reduce test pattern generation costs. These cost savings are even apparent when comparing the LSSD and non-LSSD automatic test generation results for the SCS481. The non-LSSD test generators required significantly more manual effort and wasted computer time.

As mentioned in Section 6.2, incorporating the advantages of LSSD involves significant alterations to any existing test generation program or algorithm. However, this non-recurring cost of alteration should easily pay for itself in the cost savings associated with LSSD.

As circuit densities increase, the testing problem becomes even more difficult. Not only do more faults exist on any one chip, but the inadequacies of the stuck fault model begin to become apparent. As new fault models are created to account for these "nearest neighbor" interactions, the number of faults which must be tested expands exponentially with circuit density. When this situation occurs, an LSSD-like approach will probably be the only viable testing alternative.

7.8 LIFE CYCLE COST

Extending the LSSD concept to successively higher packaging levels in an equipment or system design should result in significant Life Cycle Cost savings. Life Cycle Cost (LCC) is defined as the sum of development, acquisition and operating and support (O&S) costs. Typically the LCC elements break out as follows:

- (a) Development Costs - 5% to 15% of total LCC
- (b) Acquisition Costs - 30% to 60% of total LCC
- (c) Operating and Support Costs - 35% to 65% of total LCC

While implementation of LSSD may add to development and acquisition costs, this need not be true. In commercial applications of LSSD we have found that the valuable LSSD attributes of program-generated test data, high test coverage, and the ability to observe and control every latch have resulted in reduced cost of both development and manufacturing. In addition, the improved fault detection and isolation capability will reduce operating and support costs in military applications by accomplishing the following:

- (a) Reducing spares required at operating sites and in repair pipelines due to a reduction in false removal rates (i.e., removal of good modules because of poor diagnostic resolution).
- (b) Reducing maintenance training requirements at all levels of maintenance, but especially at the organizational level where maintenance skills are low and turn-over rates are high.

7.9 STANDARDIZATION

LSSD methodology, through its design rules, encourages standardization of design architectures related to latch and clocking and latch interconnection. This allows LSSD test generators to yield their optimum results. Standardization is also seen in the hardware required for testing LSSD devices. A tester capable of exercising the Shift Register Input, Shift

Register Output, A Clock, and B Clock lines along with the other Primary Inputs/Outputs can be used for testing any LSSD integrated circuit or higher level assembly.

7.10 MAINTENANCE CONCEPTS

The implementation of LSSD offers the potential for a more highly automated maintenance concept with less reliance on maintenance technician skills. For military applications, skill levels are in extremely short supply at the organizational and, to a slightly lesser extent, the intermediate or shop maintenance level. The power of LSSD as a testing tool, together with the size and reliability advantages of VLSI offers the potential for fault tolerant design. Fault tolerance can permit deferral of required maintenance actions for performance at a centralized maintenance facility where higher skills are concentrated.

In addition, LSSD can be used to provide an integrated test approach from chip level to system level. Historically, fault detection and diagnosis features are developed for each maintenance level. Correlation of test results between maintenance levels has, at best, been a costly and difficult process. With LSSD, the chip level shift registers are essentially connected into successively longer shift registers at each higher packaging level. Finally, at the system level, the LSSD technique is used to provide system checkout each time the system is activated, or when necessary for fault detection and diagnosis by maintenance personnel. In this way chip level tests bear a known relationship to system level tests and to fault isolation tests performed at each physical packaging level.

7.11 OPERATIONAL AVAILABILITY

Operational availability is defined as the probability that a system or equipment, when used under stated conditions and in an actual logistics

environment, shall operate satisfactorily at any given time. It may be expressed as:

$$A_o = \frac{MTBF}{MTBF + MTTR + MLDT}$$

where MTBF = mean time between failures

MTTR = mean time to repair, given that a spare is available at the repair site

MLDT = mean logistics delay time to obtain a spare from a supply remote from the repair site

The high effectiveness of the LSSD technique, where nearly 100% of the total faults are tested, allows for a much lower MTTR. Conventional testing may not reveal a fault resulting in manual intervention by a technician to repair the machine. The lower MTTR for LSSD results in a higher operational availability. The net effect of the LSSD technique is to provide high operational availability for a system while reducing support costs through reduction in quantities and skill levels of required maintenance technicians and reduction in spares quantities through centralization of maintenance facilities.

7.12 RELIABILITY

The reliability of a component is defined as the survival probability of that device (the probability that the device will survive beyond time t). Since the survival of a device is based on the technology from which it was manufactured, LSSD design and test techniques will have no effect on the device's reliability.

SECTION 8
IBM CONCLUSIONS AND RECOMMENDATIONS
WITH RESPECT TO LSSD IN THE MIL ENVIRONMENT

8.1 CONCLUSIONS

8.1.1 LSSD/NON-LSSD DESIGN

Implementation of the LSSD rules was found to require 3.5% additional logic gates and 6.5% additional I/O pins when comparing the LSSD SCS481 to the non-LSSD SCS481 with off chip clock generation.

LSSD was found to have approximately the same impact on layout complexity and power consumption. LSSD performance impacts were found to be negligible.

Replacing the least significant position 74S481 in the AP-101C with the LSSD SCS481 and passing the AP-101C Factory Test Program demonstrated that a level sensitive, LSSD design could function transparently in an edge-triggered environment. This was significant since most near term military applications of LSI/VLSI must still interface with edge-triggered vendor logic.

LSSD was found to enhance maintenance procedures and, therefore, improve operational availability because of its significant increase in testability. This improvement in maintainability contributed to a decrease in life cycle costs for products designed with LSSD. However, the most significant advantage attributed to LSSD was its effect on test capability which is discussed in the next section.

8.1.2 LSSD/NON-LSSD TEST CAPABILITY

LSSD was found to increase test capability when compared to non-LSSD designs in several significant areas. The level sensitive nature of LSSD

guarantees test pattern integrity eliminating wasted time debugging faulty patterns. The scan feature of LSSD allows all test generation to be combinational in nature greatly simplifying the automatic test generation effort. Strictly combinational testing allows test patterns to be automatically subsumed decreasing test data volume and fault simulation time.

The scan feature of LSSD also enhances design partitioning by allowing the SRL's to be treated as additional boundaries. Design partitioning has the ability to significantly decrease test generation costs. Importantly, all the LSSD advantages can be extended beyond the first level package to include the entire system, provided LSSD groundrules are observed.

In order to gain full advantage of the LSSD concepts, some changes to CAD support and hardware testers are required. However, the advantages of LSSD for LSI/VLSI test capability and maintainability, far outweigh the alterations required for its support.

8.2 RECOMMENDATIONS

- (a) Upon the completion of this phase of the contract, it appears advantageous to adopt LSSD design rules for military LSI/VLSI designs. LSSD rules for testing internal arrays should also be adopted to provide further flexibility and testability. The adoption of these guidelines for all future military LSI/VLSI designs will significantly enhance the testability and maintainability of these designs.
- (b) A further recommendation is that a study should be undertaken to determine if LSSD concepts can be altered successfully for application to edge-triggered vendor SSI/MSI TTL designs at the second packaging level. Any near term military designs will still incorporate significant quantities of this type of non-LSSD logic; therefore, some method of improving the testability and maintainability of these second level packages should be developed. Conceptually, LSSD concepts can accomplish this goal.

(c) Lastly the military must be the driving force in establishing increased testability and maintainability for their systems. Provisions for the additional hardware, software, and development time necessary to incorporate sound testability and maintainability concepts into military systems must be made in any future contracts. Without this emphasis from the military, the contractors are unlikely to include these features of their own accord. Adoption of LSSD design rules for future designs can provide an excellent starting point for improving system testability and maintainability.

APPENDIX A
SCS481/74S481 DIFFERENCES

Although the SCS481 performs the functions of the 74S481, minor differences exist between the two parts since different technologies, packaging and designs were used for their implementation. These differences exist in the following areas:

- (a) Slice Position Input
- (b) Pinout
- (c) Package
- (d) Power Supply Requirements
- (e) Electrical Requirements
- (f) Critical Path Performance
- (g) SCS481 Enhancements
- (h) SCS481 Test Circuits
- (i) SCS481 Logic Design Errors and Corrections

A.1 SLICE POSITION INPUT

Operation of the 74S/SCS481 is different depending on the location of the device in the hierarchy of the system. This location, least significant position (LSP), intermediate position (IP), or most significant position (MSP), is determined by the analog level on a single pin in the 74S481. The position input on the SCS481 is described digitally on two pins because the necessary analog comparator circuits were not available in MS399.

74S481

- (a) PIN 19 AT 0V -- IP SLICE
- (b) PIN 19 AT 2.4V -- LSP SLICE
- (c) PIN 19 AT 5V -- MSP SLICE

SCS481

- (a) PINS D2, L9 = 00, MSP SLICE
- (b) PINS D2, L9 = 11, LSP SLICE
- (c) ELSE = , IP SLICE

A.2 SCS481/74S481 PINOUT COMPARISON

Table A.2

SCS481 PIN NUMBER	74S481 PIN NUMBER	PIN NAME	PIN FUNCTION	INPUT, OUTPUT, OR INPUT/OUTPUT
H1	35	INC/MC	When low, enables the MC to increment as directed by \overline{CCI} on the next clock transition. When high, inhibits MC to hold mode. As \overline{CCO} is common to MC and PC, the MC should be inhibited when PC is enabled.	Input
G4, H4, H3 D9, D8, E8, E7, G5	36	GND	Common or ground terminal for the supply voltage.	
B4, A4 B10, C4	46, 47 1, 2	BI/00, BI/01 BI/02, BI/03	4-bit parallel data input port to the B latch, or 4-bit parallel data output for the Σ -bus when not being used as an input.	Inputs/Outputs Input Only for SCS
C2, D4 D1, C3	6, 5 4, 3	A10, A11 A12, A13	4-bit parallel data input port to the A latch and WR.	Inputs
H2, F4 G2, E2 D3, G1 E1, K7 L3, E4 (J6)	7, 8 9, 10 17, 14 13, 11 15, 16	OP0, OP1 OP2, OP3 OP4, OP5 OP6, OP7 OP8, OP9	OP0 through OP9 serve as a 10-bit parallel operation-select input to the micro-decode logic array. In the most-significant position, OP8 and OP9 additionally serve as open-collector outputs during multiply and divide algorithms. In the least-significant position, OP9 serves as an open-collector output during the CRC algorithm.	Inputs
D7, H5	12	V _{CC}	5-volt power-supply terminal	Supply Voltage Pin
G3	18	\overline{CIN}	Receives low-active ripple carry input data.	Input
D2, L9 SEE SCSP05	19	POS	Directs internal and input/output end-conditions required to define the relative position of each bit-slice when N-SN74S481's are cascaded to implement Nx4-bit word lengths. When biased at 2.4 volts, the package operates as the least-significant (LSP) slice; when grounded, it functions as an intermediate (IP) slice; and when high, 5 volts, it functions as the most-significant (MSP) slice.	Input
K5	20	Y/AG	In least-significant and intermediate positions outputs arithmetic carry generate (Y) for use with look-ahead. In most-significant position outputs true arithmetically-greater-than signal.	Output
L5	21	X/LG	In least-significant and intermediate outputs arithmetic carry propagate (X) for use with look-ahead. In most-significant position outputs true logically-greater-than signal.	Output

Table A.2 (continued)

SCS481 PIN NUMBER	74S481 PIN NUMBER	PIN NAME	PIN FUNCTION	INPUT, OUTPUT, OR INPUT/OUTPUT
B6, A5 D6, B7	38, 39 40, 41	AOP0, AOP1 AOP2, AOP3	4-bit parallel address-out port.	Outputs
A3	42	A0	Selects one of two AOP sources (PC or MC).	Input
C1	43	INCPC	When low, enables the PC to increment as directed by CCI on the next + clock transition. When high, inhibits PC to hold mode. As CCO is common to PC and MC, the PC should be inhibited when MC is enabled.	Input
J1	44	CCI	In least-significant position, a low input directs enabled PC or enabled MC to increment by one on next + clock transition. In the LSP, a high directs enabled PC or enabled MC to increment by 2. In other positions, a low is a carry input and a high inhibits the counter.	Input
H11, L2	45	CK	When high, enables the transparency of A and B input latches. When low, latches A and B input data. Clocks synchronous registers and counters on the positive transition.	Input
-	48	BI/O SEL	When low, enables BI/O to output Σ -bus data. When high, the BI/O output drivers are placed in a high-impedance state permitting BI/O to be used as data inputs.	Input
E5, S7		V _C	1.7V Power Supply	Supply Voltage Pwr.
D2, L9		SCSP0S	00 = MSP, 11 = LSP, 10 or 01 = IP	Inputs
B9			B Clock Output	Output
E2	37	CCO/OV	In least-significant and intermediate positions a low-level output indicates that either the PC or MC is at maximum count. As CCO is common for both PC and MC ambiguous carry can be avoided if one or both counters is/are disabled by the INCPC and/or INCMC inputs. In the most-significant position, a high-level output, depending on the operation selected, indicates that the WR, XWR, or ALU will overflow (OV) on the next clock.	Output

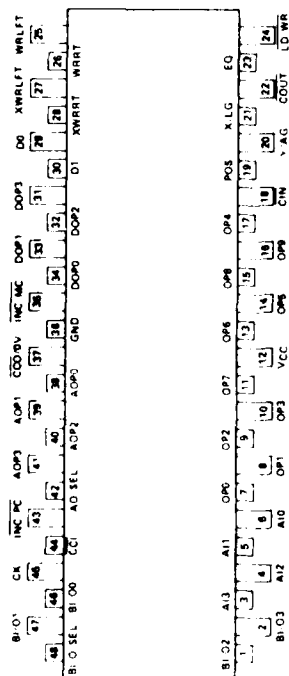
Table A.2 (continued)

SCS481 PIN NUMBER	74S481 PIN NUMBER	PIN NAME	PIN FUNCTION	INPUT, OUTPUT, OR INPUT/OUTPUT
H6	22	$\overline{\text{COUT}}$	Outputs low-active ripple carry data.	Output
J5	23	EQ	Outputs true (active-high) equality of γ' bus equals zero for each 4-bit slice. The open-collector output permits wire-AND to achieve Nx4-bit equality output.	Open-Collector Output
B1	24	$\overline{\text{DWR}}$	When low, data applied at the A1 port coincident with the \pm clock transition is loaded into the WR.	Input
F9 K4	26 25	WRRT, WR_FT	Working register and Σ -bus shift interconnectivity pins. WRRT receives left-shift and output right-shift (true) data. WRFLT receives right-shift and outputs left-shift (true) data. Shift can be single-precision, double-precision, signed or unsigned.	Bidirectional I/O SCS Open Collector
G10 L4	28 27	XWRRT, XWRLFT	Extended working register shift interconnectivity pins. XWRRT receives left-shift and outputs right-shift (true) data. XWRLFT receives right-shift and outputs left-shift (true) data. Shift can be single-precision, double-precision, signed or unsigned.	Bidirectional I/O SCS Open Collector
H10 J11	29 30	D0 D1	Selects one of three DOP sources (WR, XWR, or Σ -bus) or places the DOP outputs in a high-impedance state.	Inputs
F11 F10 H9 G11	34 33 32 31	DOP0 DOP1 DOP2 DOP3	4-bit parallel, data-out port, DOP0 is LSB.	3-state outputs (No 3-state for SCS)
C5, K10 K9, J8		BP0, BP1 BP2, BP3	4-bit parallel data input port to the breakpoint register	Inputs
K8		PSH	When Active with ENSK pushes data onto stack coincident with \pm clock transition. Complement pops data off stack	Input
B2		LMC	Loads A-mux from MC for any OP code	Input
D11, C11		ENRO0, ENRO1	Complement enables ring oscillator	Inputs
E9, D5 C10, C9		TST0, TST1 TST2, TST3	Selects Breakpoint Register or test data for the BP1ST port	Inputs

Table A.2 (continued)

SCS481 PIN NUMBER	74S481 PIN NUMBER	PIN NAME	PIN FUNCTION	INPUT, OUTPUT, OR INPUT/OUTPUT
F1, F2, F3, F4		B02, S01 B02, S03	B Bus Out Bits 0,1,2,3 4 Bit parallel data output for the E-bus.	Outputs
G5, L6		QF90, QP90A	QF90 Bit 9 Out for multiply, divide and CRC algorithms (open collector)	Outputs
B5		PCEQ	Outputs true when PC equals BP for each 4-bit slice. The open collector output permits wire-AND	Output
C6		MCEQ	Outputs true when MC equals BP for each 4-bit slice. The open collector output permits wire-AND	Output
A9		S0	Scan Out/Ring Oscillator position 1 Out	Output
B8		R0	Ring Oscillator position 14 Out	Output
G8		DCPHZ	Outputs true when D0 and D1 both equal one	Output
A7, A6 A8, C7		BPTST0, BPTST1, BPTST2, BPTST3	Outputs Breakpoint Register or test signals determined by test select inputs TST0-3	Outputs
C8			Test CEE0 Out	Output
G9		HLDA6	Disables clock to both A & B Latches	Input
J8		ECLK	LSSD Test B Clock (Complement)	Input
J9		CCLK	LSSD Test C Clock (Complement)	Input
E10		HLDA	Disables Clock to Latch A	Input
F8		HLDB	Disables Clock to Latch B	Input
K11		ACK	LSSD Test A Clock (True)	Input
D10		SI	Scan In	Input
B3		LSKPC	Load PC from Stack	Input
J10		LBR	Load Breakpoint Register	Input
E11		ENSK	Enable Stack	Input

A.3 SCS481/74S481 PACKAGE COMPARISON



74S481

A	B	C	D	E	F	G	H	J	K	L	Bevel Edge
			S05	R04	R02	R01	Q01	N02	M02	L02	11
	R10	R06	S04	S03	S01	P02	N01	N01	G02	F01	10
S12	S11	S10	GND	R05	S02	Q02	N01	N01	L01	F02	9
R13	R12	R11	GND	GND	R03	P01	G01	G01	E01	D01	8
S15	R14	S13	VCC	GND		VC	C01	C01	C02	B01	7
R15	S17	S16	S14				A04	A04	A02	A01	6
R16	R17	Q16	Q17	VC			GND	VCC	A05	A03	5
P16	P17	N17	L17	D17	B15	B15	GND	GND	B07	B06	4
N16	M16	G17	E17	C16	A16	B13	GND	GND	A08	A07	3
	M17	L16	F17	D16	A17	A15	A14	A14	B12	B08	2
	G16	F16	E16	C17	B17	B16	B14	B14	A13		1

SCS481

Figure A.3 74S/SCS481 Package Comparison

A.4 POWER SUPPLY REQUIREMENTS

The SCS481 dissipates 1.4 watts of power versus 1.9 watts for the 74S481. The following line items describe the power supply requirements:

74S481

- (a) + 5V \pm 5% AT 380 mA TYP
- (b) 1.9 WATTS TYP

SCS481

- (a) + 5V \pm 10% AT 53 mA TYP
- (b) + 1.7V \pm 5% AT 670 mA TYP
- (c) 1.4 WATTS TYP.

A.5 SCS481/74S481 ELECTRICAL REQUIREMENTS

Table A.5

		74S481		SCS481	
PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH} min-level output voltage	Except EQ9	2			V
V_{IL} low-level input voltage	Except EQ9				V
V_{I} input clamp voltage	$V_{CC}=MIN, I_I=-10\mu A$			-1.2	V
V_{OH} min-level output voltage	Any I/O or output except EQ, OP8, OP9	$V_{CC}=MIN, V_{IH}=2V, V_{IL}=0.8V, I_{OH}=MAX$	2.5		V
		74S	2.7	3.4	
V_{OL} low-level output voltage	$V_{CC}=MIN, V_{IH}=2V, V_{IL}=0.8V, I_{OL}=MAX$			0.5	V
I_{OH} min-level output current	EQ, OP8, OP9			100	μA
I_{I} input current at maximum input voltage	POS			1	nA
	Any other input			1	nA
I_{IH} min-level input current	OP0, OP1, OP2, OP3, CIN			200	μA
	Any other except POS			100	μA
I_{IL} low-level input current	OP0 thru OP3, CIN, POS, CCI, WRRT, WRFT, XWRRT, XWFT, CLOCK			-8	nA
	Any other input			-6	nA
	Any other input			-2	nA
I_{OP} short-circuit output current	Any output or I/O except EQ, OP8, OP9	$V_{CC}=MAX$	-30	-100	mA
I_{IO} standby current		$V_{CC}=MAX$	380	425	nA
I_C supply current		$V_C=MAX$			mA

* SCS481 PING E4 (OP9), E9 (TEST SELECT 0), AND H11 (CK) HAVE MAXIMUM LOW LEVEL INPUT CURRENT OF -2.4 mA.

A.6 CRITICAL PATH PERFORMANCE COMPARISON

Table A.6

DELAY SIMULATION TIMING ANALYSIS KEY CRITICAL PATHS

PATH	TI SPECS		DELAY SIMULATION *	
	TYP	MAX	NOMINAL	WORST CASE
MULTIPLY OP9#	45NS	90NS	32.4NS	48.6NS
DIVIDE OP9 (FIX) α	45NS	90NS	35.5NS	53.3NS
A/B BUS TO X.Y	32NS	60NS	28.3NS	42.5NS
A/B BUS TO AG,LG	60NS	100NS	35.9NS	53.9NS
A/B BUS TO EQUAL	45NS	75NS	30.4NS	45.6NS
A/B BUS TO DATAOUT %	42NS	75NS	38.4NS	57.6NS
A/B BUS TO OVERFLOW	35NS	60NS	31.5NS	47.3NS
CI TO COUT	30NS	50NS	17.8NS	26.7NS
OPBUS TO DOP \$	70NS	115NS	40.8NS	61.2NS

* OFF CHIP DRIVERS ARE ASSUMED TO HAVE A NOMINAL DELAY OF 5NS.

THIS PATH CONTAINS 2 OCD'S AND SOME INTERCHIP DELAY ASSUMED EQUAL TO ZERO.

α THIS PATH IS DETERMINED BY \overline{CIN} PROPAGATION. THESE FIGURES ARE FOR ONE SCS481 ONLY.

\$ THIS PATH IS FOR OP FORM 1 ONLY

A.7 SCS481 ENHANCEMENTS

Unused cells and logic I/O in the MS399 gate array after the initial SCS481 design allowed six enhancements to be added to the 74S481 functions. Functional block diagrams of the enhancements follow. These enhancements are documented in Section 3.1.4:

- (a) Bus from MC to A-Mux
- (b) Gate the clock to the A,B Registers
- (c) 4 x 4 Stack for the PC
- (d) Breakpoint Register
- (e) Dual OP9 outputs
- (f) B-Bus Out Separated from Input

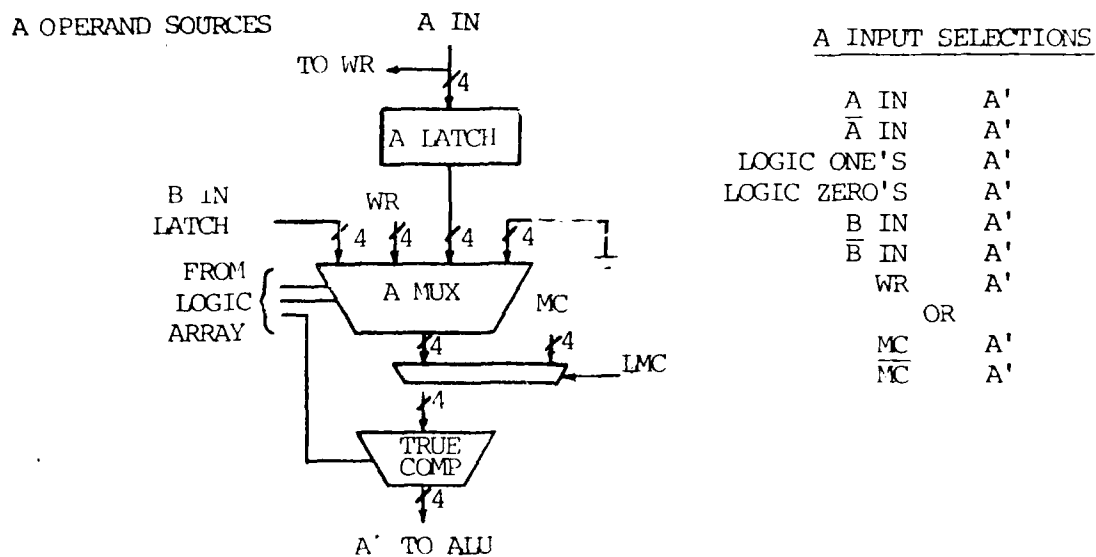


Figure A.7-1 MC to A-Mux Enhancement

AD-A135 899

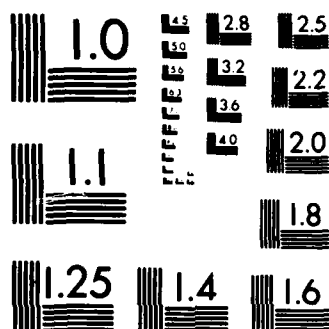
LSI (LARGE SCALE INTEGRATED) DESIGN FOR TESTABILITY
FINAL REPORT OF DESIG. (U) IBM FEDERAL SYSTEMS DIV
MANASSAS VA R D GROVES ET AL. NOV 83 AFWAL-TR-81-1068
F/G 5/1

4/4

UNCLASSIFIED

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

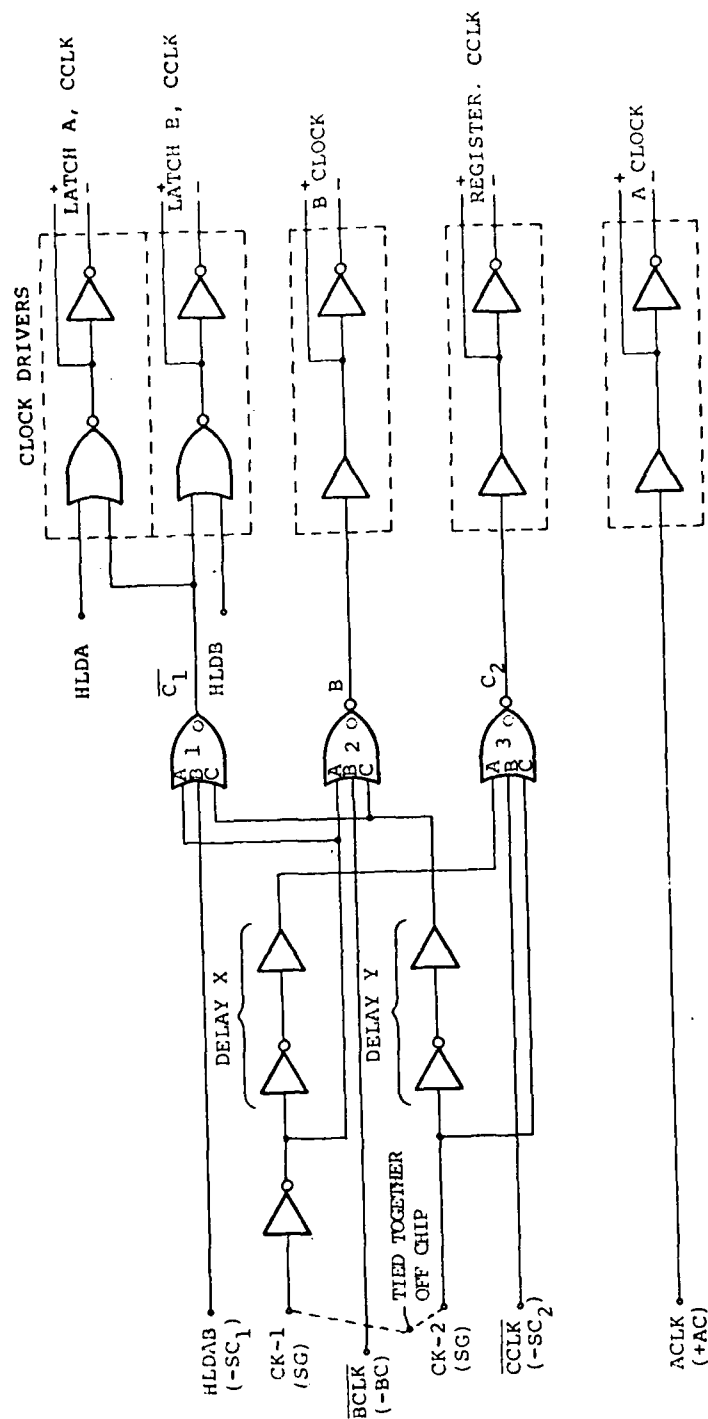


Figure A.7-2 Gated A and B Register Clocks

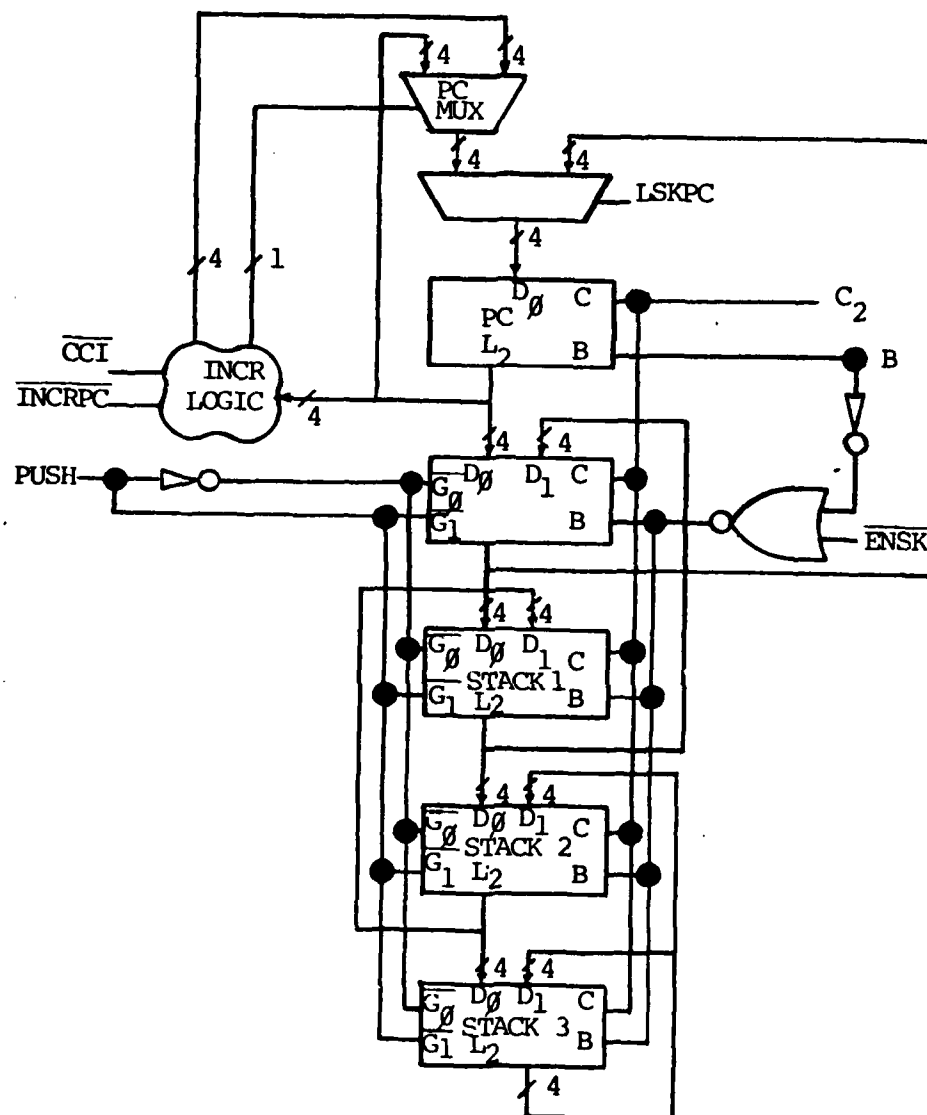


Figure A.7-3 4x4 Stack for PC

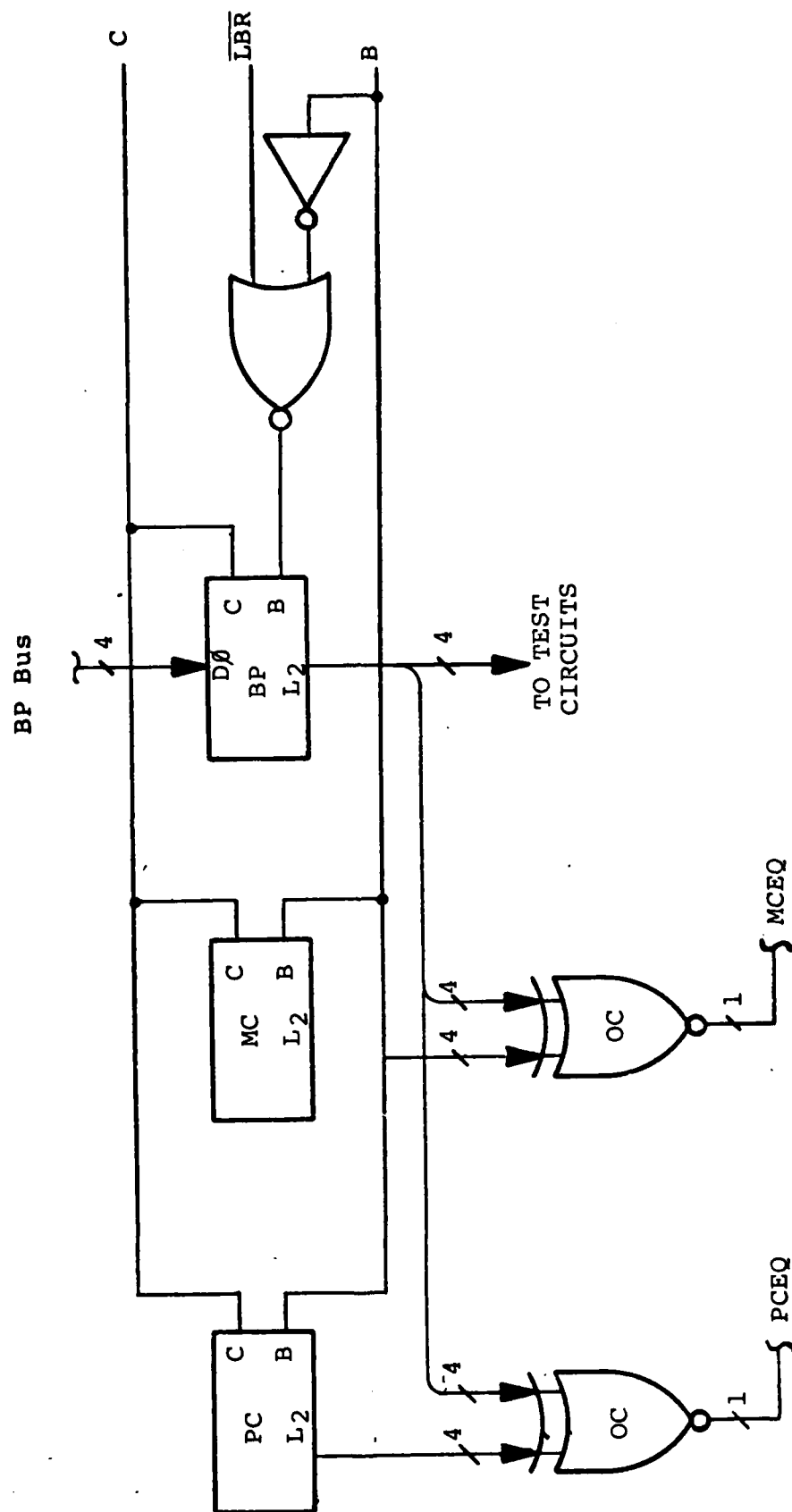


Figure A.7-4 Breakpoint Register

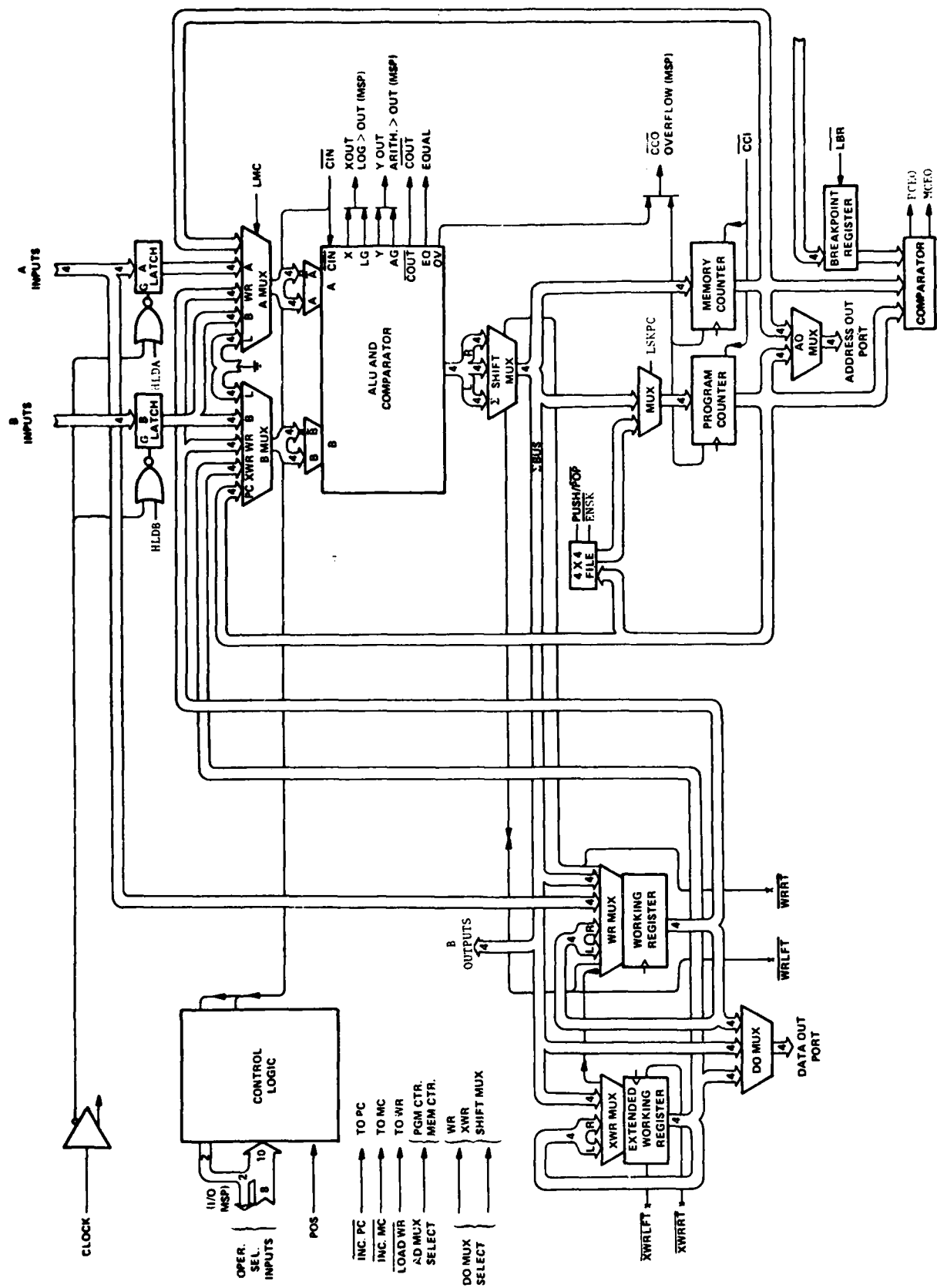


Figure A.7-5 SCS481 Functional Block Diagram

A.8 SCS481 TEST CIRCUITS

Since extra logic and extra I/O pins existed on the SCS481 after the logic design of the 74S481 functions, four basic test circuits were added. These four circuits are described in Section 3.1.5:

- (a) Ring Oscillator
- (b) DOP Three-state Control
- (c) CEE0 Test Circuit
- (d) CEC8/CEC0/CED8/CED7 Test Circuit.

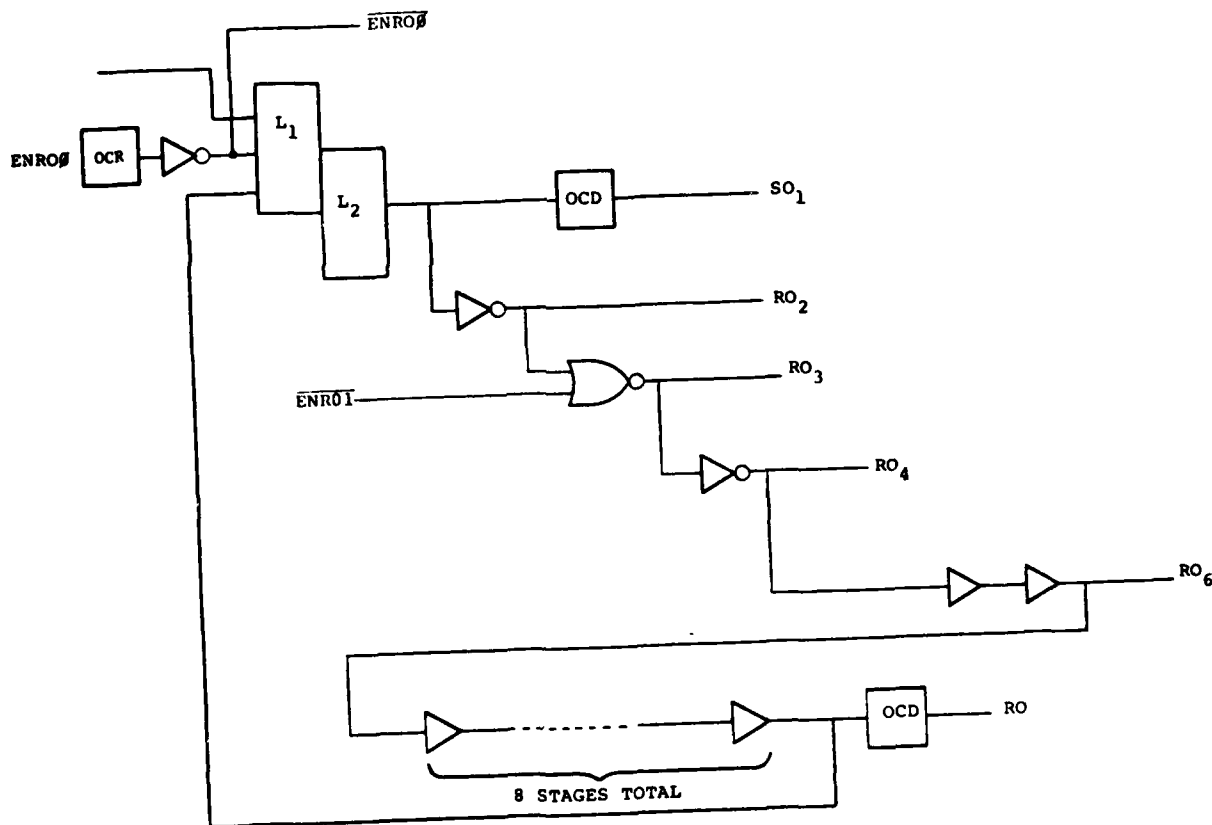


Figure A.8-1 Ring Oscillator

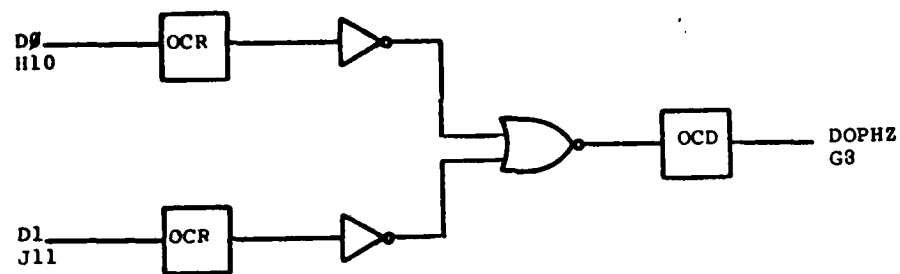


Figure A.8-2 DOP Three-State Control

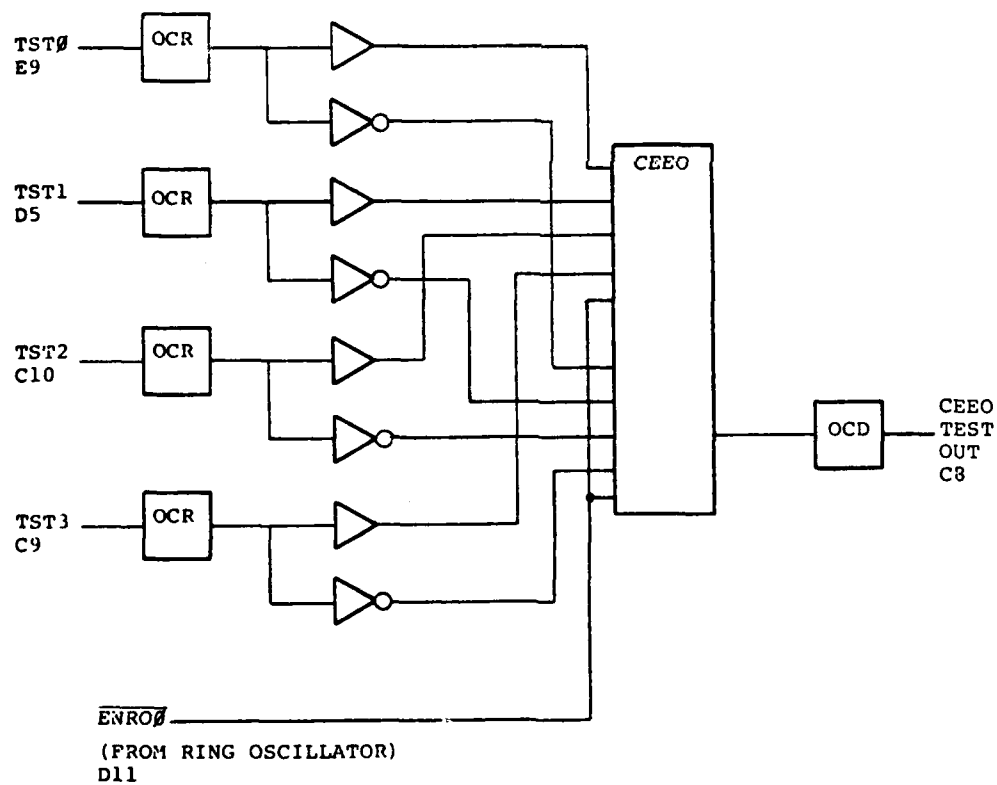
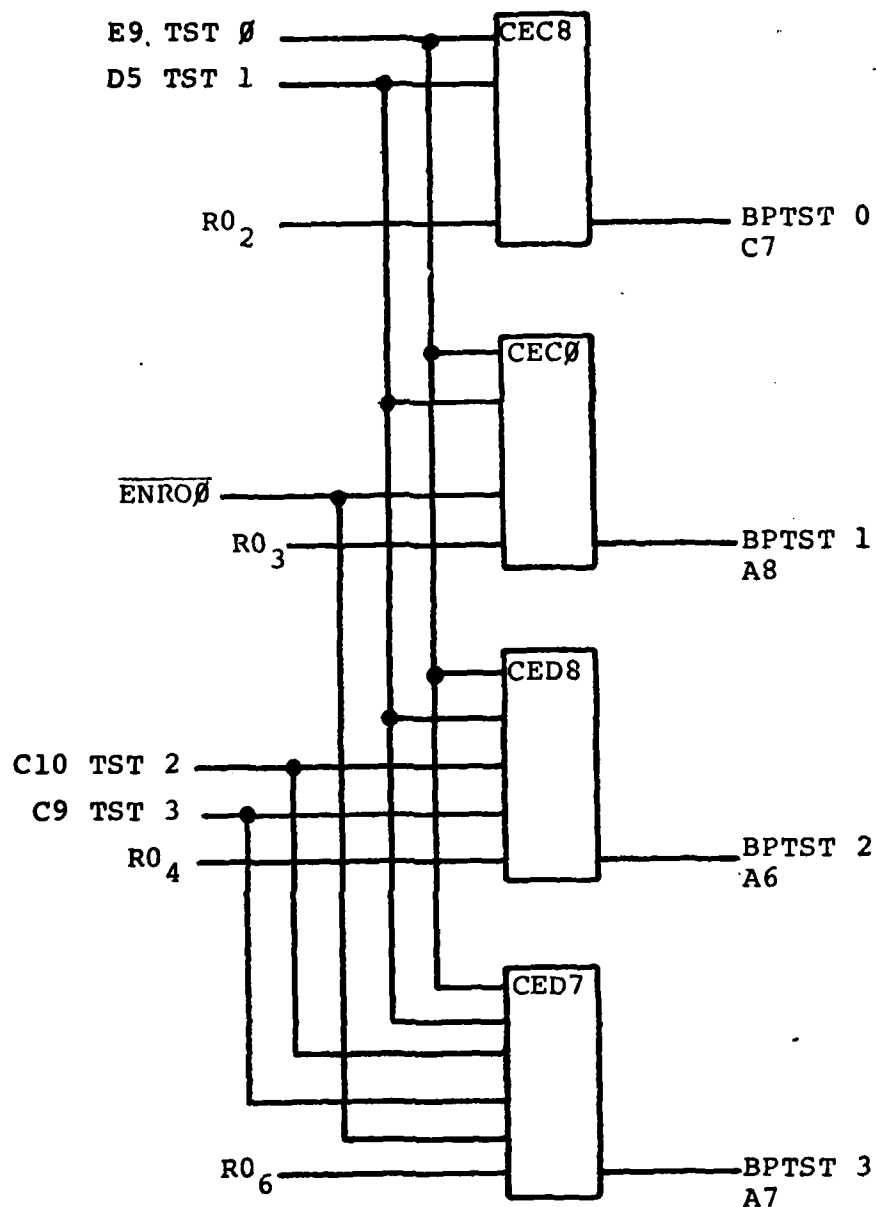


Figure A.8-3 CEE0 Test Circuit



TEST SELECT		BP/TEST OUT			
0	1	0	1	2	3
0	0	BP ₀	BP ₁	BP ₂	BP ₃
0	1	BP ₃	BP ₀	BP ₁	BP ₂
1	0	BP ₁	BP ₂	BP ₃	BP ₀
1	1	R0 ₂	R0 ₃	R0 ₄	R0 ₆

Figure A.8-4 CEC8/CEC0/CED8/CED7 Test Circuit

A.9 SCS481 LOGIC DESIGN ERRORS AND CORRECTIONS

Logic design errors resulted in five I/O lines being logically inverted from their function in a 74S481. Those lines are XWRRT, WRRRT, XWRLFT, WRLFT, and CCI (inverted for LSP). The circuits required to correct these errors are shown in the following figure. Please note that these circuits are required only when substituting an SCS481 for a 74S481. When only SCS481's are used in the circuit, the correction circuits are not required.

LM
1-8